UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Maiquel de Brito

UMA LINGUAGEM PARA ESPECIFICAÇÃO DA DINÂMICA DOS FATOS INSTITUCIONAIS EM SISTEMAS MULTIAGENTES

Florianópolis

2012

Maiquel de Brito

UMA LINGUAGEM PARA ESPECIFICAÇÃO DA DINÂMICA DOS FATOS INSTITUCIONAIS EM SISTEMAS MULTIAGENTES

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.
Orientador: Prof. Dr. Jomi Fred Hübner

Florianópolis

2012

Catalogação na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

B8621 Brito, Maiquel de
Uma linguagem para especificação da dinâmica dos fatos institucionais em sistemas multiagentes [dissertação] / Maiquel de Brito ; orientador, Jomi Fred Hübner. – Florianópolis, SC, 2012.
147 p.: il., tabs.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de sistemas. 2. Sistemas multiagentes. 3. Linguagem de programação (Computadores). I. Hubner, Jomi Fred. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

CDU 621.3-231.2(021)

Maiquel de Brito

UMA LINGUAGEM PARA ESPECIFICAÇÃO DA DINÂMICA DOS FATOS INSTITUCIONAIS EM SISTEMAS MULTIAGENTES

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 30 de março 2012.

Prof. Dr. José Eduardo Ribeiro Cury
Coordenador do Curso

Prof. Dr. Jomi Fred Hübner
Orientador

Banca Examinadora:

Prof. Dr. Jomi Fred Hübner
Presidente

Prof. Dr. Ricardo José Rabelo

Prof. Dr. Luis Otávio Alvares

Prof. Dr. Antônio Carlos da Rocha Costa

AGRADECIMENTOS

Inicialmente, presto profundo agradecimento ao Prof. Jomi Fred Hübner por ter me orientado ao longo do desenvolvimento desta dissertação. Agradeço por sua constante confiança, dedicação, interesse, respeito e valor prestados ao trabalho. Pelas horas empenhadas em pensar, divagar e debater. Pelo acolhimento às boas ideias e pelos respeitosos alertas sobre os perigosos equívocos. Agradeço pelas orientações que não se limitaram a essa dissertação, mas que se estenderam à minha formação como pesquisador em diversos aspectos.

Agradeço à Profª Carine Webber por ter me encorajado a ingressar no mestrado e aos professores e pesquisadores Alessandro Ricci, Michele Piunti e Rafael Bordini por suas opiniões, sugestões, avaliações e críticas durante a realização do trabalho. Agradeço também à CAPES pelo apoio financeiro através da bolsa de mestrado.

Em função de o mestrado ter me conduzido a morar em uma nova cidade, agradeço aos amigos de longos anos que deixei em minha terra de origem e aos amigos que tive a oportunidade de conhecer em Florianópolis. Da mesma forma, agradeço familiares pelo constante apoio. Com isso me refiro tanto aos familiares de quem me distanciei para poder ingressar no mestrado quanto àqueles que acabaram por se tornar minha família em Florianópolis. E, em especial, a Cinthia, que, durante o desenvolvimento deste trabalho, foi namorada, noiva e esposa, por todo apoio, incentivo, confiança e compreensão.

Por fim, agradeço a Deus por ter me concedido cada um dos dias que foram necessários à realização desse trabalho.

Agora, pois, vemos apenas um reflexo obscuro, como em um espelho; mas, então, veremos face a face

São Paulo.

RESUMO

Esta dissertação foi desenvolvida considerando que Sistemas Multiagentes (SMA) são compostos por três dimensões distintas e independentes: agentes, ambiente e instituição. Os agentes podem atuar sobre o ambiente e a instituição alterando o estado dessas estruturas. Este trabalho concentra-se especificamente em fatos que provocam alterações no estado da instituição, que, neste trabalho, são chamados *fatos institucionais*.

Os fatos institucionais podem ser resultado da ação direta dos agentes. Um agente pode, por exemplo, interagir com a instituição para adotar um determinado papel ou para comunicar que cumpriu uma obrigação que lhe fora atribuída. Em outras situações, no entanto, o estado da instituição pode ser alterado a partir de fatos verificados no ambiente ou na própria instituIção. Essa abordagem tem inspiração na teoria da Construção da Realidade Social, proposta por John Searle. Um agente cruzando um semáforo fechado, por exemplo, é um fato ocorrido no ambiente e que tem uma consequência em nível institucional, que é a violação de uma norma. Esse agente, no entanto, pode não ter intenção em comunicar a violação à instituição pois isso pode ter consequências contrárias a seus objetivos. Nesse caso, é conveniente que a instituição identifique a violação da norma a partir do fato ocorrido no ambiente, independente da vontade do agente.

A partir dessas questões, este trabalho propõe um modelo para definir alterações no estado da instituição a partir de eventos e estados verificados no ambiente ou na própria instituição. Com base no modelo proposto, propõe-se também linguagem de programação para escrever regras que definem alterações no estado da instituição e uma arquitetura que foi implementada. Por fim, para avaliar o modelo proposto, a implementação realizada é incorporada em dois SMA. Como resultado desses estudos, observou-se a possibilidade de simplificação no raciocínio e atuação dos agentes, que podem concentrar-se em atuar sobre elementos do ambiente sem conhecer integralmente as instituições com que se relacionam na realização de suas tarefas. Verificou-se também que o modelo proposto pode contribuir com a persistência do SMA, além de conferir-lhe a característica do poder institucional.

Palavras-chave: Sistemas Multiagentes, Instituição, Ambiente, *constitutive rules*

ABSTRACT

This work was writen regarding Multiagent Systems as composed by three distinct and independent dimensions: agents, institution and environment. The agents can act over the environment and institution causing changes on these structures. The state of the institution can, moreover, be changed as result of facts that have occurred in the environment or even in the institution. This approach is inspired on the theory of Construction of Social Reality, proposed by John Searle. For instance an agent running through a red traffic light means a norm violation. In this case, the fact of an agent running through a red traffic light is an action of the agent over the environment and the norm violation is the new institutional state.

Up on the basis on this approach is proposed a model to state changes on the state of the institution as consequence of facts occurred in the environment or even in the institution. Based on the model, it is also proposed a programming language and an architecture that was implemented. Finally, in order to evaluate the model, the implementation is inserted in two experimental Multiagent Systems. As results, it was possible to observe the possibility of simplification on the agent's reasoning and acting. The agents can focus on acting over environment elements without any awareness about the institutional infrastructure of the system. The model can also contribute to the system persistence and give to the system the feature of institutionalised power.

Keywords: Multiagent systems. Institution. Environment. Constitutive Rules.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

Este trabalho trata de Sistemas Multiagentes (SMA) e, mais especificamente, de SMA abertos, em que agentes podem entrar e sair livremente (DEMAZEAU; COSTA, 1996). Uma importante característica de SMA abertos é a impossibilidade de se conhecer integralmente, em tempo de projeto, o número, o comportamento e a forma como os agentes irão interagir com os recursos do sistema (PIUNTI, 2009). Como consequência, estes sistemas podem apresentar heterogeneidade de agentes, conflitos entre interesses indiviuais dos agentes, confiabilidade limitada e inconformidade com as especificações (ARTIKIS; PITT; SERGOT, 2002). Diante desses aspectos, torna-se necessário o estabelecimento de mecanismos que conciliem a atuação autônoma dos agentes com a manutenção do funcionamento do sistema em conformidade com os propósitos para os quais foi projetado. Uma forma de conciliar estes aspectos é através da utilização de uma dimensão institucional (ESTEVA et al., 2004).

Outro aspecto a ser considerado ao se tratar de SMA abertos é o *ambiente*. Existem duas perspectivas principais a respeito do ambiente em SMA: uma baseada na Inteligência Artificial (IA) e outra definida no contexto da Engenharia de Software Orientada a Agentes (AOSE - *Agent-Oriented Software Engineering*) (RICCI; PIUNTI; VIROLI, 2011). Segundo a noção dada pela IA, o ambiente é composto por todos os elementos pertencentes ao mundo sobre os quais os agentes podem agir e a respeito dos quais têm percepções (RUSSELL; NORVIG, 2003). Do contexto da AOSE surge a concepção, também adotada neste trabalho, de ambiente como um elemento cuja existência independe dos agentes (apesar de poder ser afetado pelos agentes), porém concebido e projetado para fornecer funcionalidades específicas aos demais elementos do SMA. Segundo essa concepção, o ambiente é considerado uma abstração primária[1].

Este trabalho considera que um SMA é composto por estas três classes de entidade: agentes, ambiente e instituição. Essas entidades, ao integrarem um SMA, não podem trabalhar de forma isolada. Ao contrário, é necessário que interajam entre si. Essas interações são ilustradas na Figura 1, sendo divididas em três categorias: interação entre agentes e instituição (A-I), interação entre agentes e ambiente (A-E - *Agents - Environment*) e interação entre instituição e ambi-

[1] Tradução livre da expressão *first-class abstraction*, que será explicada na seção 3.1.

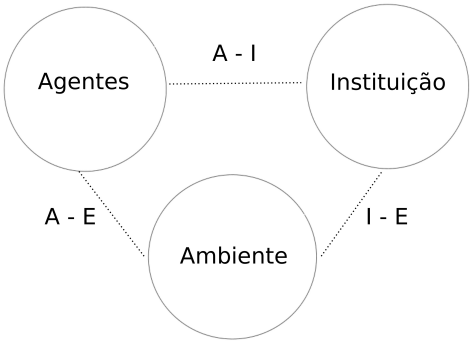


Figura 1 – Dimensões e suas interações em um SMA (adaptado de (PIUNTI et al., 2010))

ente (I-E). Para viabilizar tais interações, existem questões a serem observadas nos níveis conceitual e tecnológico. Do ponto de vista conceitual, é necessário definir maneiras de interação entre as diferentes formas de abstração. Pode-se citar como exemplo o caso da interação entre o modelo de agentes BDI (definido em termos de crenças, desejos e intenções) e o modelo organizacional $\mathcal{M}$oise (definido em termos de grupos, papeis, missões e normas) (HÜBNER; SICHMAN; BOISSIER, 2007). Nesse caso, os agentes têm mapeado, na sua base de crenças, suas obrigações, os grupos a que pertencem, os papéis que assumiram etc. Do ponto de vista tecnológico, pode-se verificar a interação entre esses elementos através da integração entre implementações baseadas no modelo $\mathcal{M}$oise (como, por exemplo, ORA4MAS (HÜBNER et al., 2009)) e a plataforma de programação de agentes *Jason*.

Para a interação entre agentes e ambiente, pode-se utilizar novamente o exemplo de agentes BDI interagindo com um ambiente projetado segundo o modelo *A&A*. Nesse caso, o ambiente é definido com base em artefatos que têm propriedades observáveis e operações. As propriedades observáveis dos artefatos são mapeadas na base de crença dos agentes e as operações são mapeadas no repertório de ações dos agentes (RICCI; PIUNTI; VIROLI, 2011). Do ponto de vista tecnológico, pode-se verificar a interação entre esses elementos através da integração entre *Jason* e o *framework* para programação de ambientes CArtAgO.

Verifica-se assim que para as interações agente-instituição e agente-ambiente existem modelos e implementações bem definidos. Quanto à interação entre ambiente e instituição, no entanto, existem aspectos conceituais e tecnológicos em aberto. Uma possível explicação

para isso é relativa novidade na concepção de ambiente como abstrações primárias (PIUNTI et al., 2010). Neste trabalho, tem-se interesse em um conjunto específico de interações entre ambiente e instituição: interações que alterem o estado da instituição a partir de fatos ocorridos no ambiente.

O interesse nesse tipo de interação justifica-se pois em determinadas situações é conveniente que o estado da instituição seja modificado independemente da atuação direta dos agentes. Em um SMA aberto, não se pode garantir que os agentes, em função de sua autonomia, terão comportamento de acordo com o esperado. Além disso, não é possível garantir que os agentes serão implementados de forma a serem capazes de interagir com a dimensão institucional do sistema. Em um sistema que implementa uma sala de aula, por exemplo, o fato de um agente entrar na sala (atuação do agente sobre o ambiente) pode implicar, em nível institucional, que esse agente passou a desempenhar o papel de *aluno*. O agente, no entanto, não necessita ter conhecimento total a respeito do aparato institucional do SMA para ter o papel de aluno, bastando-lhe apenas atuar sobre o ambiente. Além disso, em certas situações os objetivos do agente e da instituição podem ser conflitantes e a mudança do estado da instituição a partir de fatos ocorridos no ambiente pode auxiliar o sistema a se manter íntegro apesar desse conflito. Suponha-se, por exemplo, um cenário relacionado ao tráfego de veículos em uma via. Nesse cenário, um agente que passa em um semáforo fechado comete uma infração. Esse agente certamente não tem interesse em sofrer a penalidade correspondente e, por si mesmo, não irá interagir com a instituição "pedindo" que essa o penalize. Nesse caso, a interação entre ambiente e instituição pode auxiliar o sistema a tratar desse conflito de interesses, detectando a infração da norma sem necessitar da intervenção direta do agente. É possível utilizar ainda o exemplo de uma célula industrial em que robôs atuam sobre elementos do ambiente (máquinas, peças etc) para a realização de suas tarefas. Essa atuação está, certamente, inserida em um contexto institucional e deve, assim seguir certos parâmetros: as ações dos robôs devem ser feitas em uma sequência correta, por robôs corretos, sobre os elementos adequados. A atuação de um robô sobre uma determinada peça, por exemplo, é um fato ocorrido no ambiente. Essa atuação pode, no entanto, indicar o cumprimento de uma determinada obrigação (no caso de o robô ter a obrigação de fazer tal tarefa) ou a violação de uma norma (caso a tarefa tenha sido feita por um robô não autorizado, no momento errado, sobre a peça errada etc). Esses exemplos ilustram situações em que é desejável que a dimensão institucional seja afetada

por fatos ocorridos no ambiente sem necessitar da intervenção ou do conhecimento dos agentes. Na maioria das abordagens atuais, no entanto, isso depende da atuação direta dos agentes sobre a instituição.

A partir do interesse nesse tipo de interação entre ambiente e instituição, assumindo a existência de aspectos em aberto quanto a tal interação, é possível levantar algumas questões:

- Como modelar tal interação preservando a separação, tanto conceitual quanto tecnológica, entre cada uma das dimensões? que compõem o ambiente e as funcionalidades de tais elementos relacionam-se com abstrações pertencentes a infraestruturas institucionais (normas, papeis etc)?

- Como descrever tais interações através de uma linguagem de programação suficientemente expressiva?

- Como integrar tal descrição em um *framework* com infraeastruturas institucional e ambiental definidas?

Tais questões motivaram este trabalho e estabeleceram seu objetivo geral que consiste em propor um modelo de interação entre as dimensões institucional e ambiental em SMA, assumindo-as como abstrações primárias, de modo a permitir que o estado da dimensão institucional seja modificado em função de fatos ocorridos na dimensão ambiental.

O trabalho para alcançar os objetivos mencionados tem caráter exploratório e experimental. Em uma etapa inicial, será feita uma revisão sobre agentes, modelos organizacionais e ambientes em SMA, bem como um levantamento do "estado da arte" a respeito da interação instituição-ambiente. Para complementar esse estudo, busca-se inspiração na filosofia, e, especificamente, nas ideias de John Searle (SEARLE, 1995), para verificar como interações entre ambiente e instituição acontecem nas sociedades humanas. A utilização desse autor como referência justifica-se, entre outros motivos por (i) sua teoria contemplar a ideia de que o estado das instituições humanas é afetado por fatos ocorridos na realidade "bruta" (que é composta, em parte, por elementos pertencentes ao ambiente) e por (ii) sua teoria ser indicada como inspiração por outros autores da área de SMA que abordaram o assunto.

A partir do referencial levantado, o próximo passo será utilizar as informações obtidas para definir maneiras de conceber e descrever as ligações entre ambiente e instituição. Isso será feito analisando como o assunto é tratado, na filosofia, por John Searle, bem como na área de

SMA, por outros autores. O objetivo dessa análise é sintetizar como a questão tem sido abordada, quais os aspectos ainda em aberto e quais problemas enfrentados ao tratar do assunto. A partir dessa síntese, será desenvolvido um modelo para interação entre ambiente e instituição, procurando definir quais os elementos que devem compor tal modelo e como deve ser a dinâmica de tais elementos para que tal interação possibilite mudanças no estado da instituição considerando o SMA como um sistema composto por agentes, ambiente e instituição. Para avaliar o modelo quanto a sua aplicabilidade em SMA, uma das estratégias adotadas será a definição de uma linguagem formal que permita descrever as interações segundo os elementos no mesmo. Além disso, como forma de avaliar a dinâmica dos elementos presentes no modelo, será definida uma arquitetura que permita que tais elementos, a partir de uma implementação como programas de computador, consigam efetivamente, produzir os resultados esperados com a definição do modelo. Por fim, será feita uma implementação segundo a arquitetura proposta e essa implementação será incorporada SMA para a realização de experimentos. É importante ainda destacar que os passos adotados para atingir o objetivo do trabalho serão guiados pela ideia de que deve ser mantida a generalidade no modelo, permitindo que esse seja aplicado a diversos modelos de estruturas ambientais e institucionais.

Espera-se, com este trabalho, contribuir com o conhecimento na área de SMA através de um modelo suficientemente genérico e comprovadamente implementável para a especificação de alterações no estado da instituição a partir de fatos ocorridos no ambiente. Além disso, espera-se contribuir com a proposta de uma linguagem de programação para expressar tal especificação. Como "subprodutos" do desenvolvimento do trabalho, espera-se obter a definição de uma arquitetura que possa ser implementada e incorporada em SMA, bem como uma implementação da arquitetura que possa a ser incorporada a um *framework* em que o SMA seja concebido a partir das dimensões agente, ambiente e instituição.

No contexto da engenharia de automação e sistemas, tais contribuições podem ser relevantes em diversos cenários. E

O texto deste documento está estruturado da seguinte forma:

- O Capítulo 2 faz uma revisão de assuntos relacionados ao trabalho, analisando como estes são tratados na teoria de John Searle e em diversos modelos de SMA;
- O Capítulo 3 descreve o modelo proposto neste trabalho;
- O Capítulo 4 descreve uma proposta de arquitetura que viabiliza

o funcionamento do modelo proposto.

- O Capítulo 5 descreve experimentos feitos em SMAs que, originalmente, não contém mecanismos para interação entre ambiente e instituição.
- O Capítulo 6 apresenta as conclusões e propostas de trabalhos futuros resultantes do trabalho desenvolvido.

2 REVISÃO BIBLIOGRÁFICA

Muitas porções da realidade em que vivemos podem ser explicadas pelas ciências naturais por serem compostas de elementos "concretos", perceptíveis aos sentidos. Mas existem porções da realidade que são subjetivas, fruto da concordância coletiva e sustentadas por instituições, sejam elas formais ou informais. Nesse nível da realidade encontram-se as instituições que orientam e limitam a vida das pessoas em função da manutenção de certas características desejáveis nas sociedades.

Em SMA, a exemplo das sociedades humanas, também é possível utilizar estruturas com caráter institucional, como organizações e normas, para orientar e limitar a atuação autônoma dos agentes com vistas a garantir a integridade do sistema. Agentes podem influenciar e raciocinar a respeito do ambiente em que estão inseridos e também a respeito das infraestruturas institucionais. Junto com as interações agente-ambiente e agente-instituição, podem haver interações entre ambiente e instituição. Essas interações são consideradas na teoria de John Searle (SEARLE, 1995), segundo a qual fatos verificados no ambiente têm consequências em nível institucional. Em SMA, verifica-se também que alguns modelos trabalham segundo essa abordagem, admitindo que as instituições podem ter seu estado modificado através de fatos verificados no ambiente.

Este capítulo busca analisar a forma como o estado da instituição é afetado por fatos ocorridos no ambiente. Essa análise é feita, inicialmente, sob o ponto de vista das ciências sociais, verificando-se como se dão as interações dimensões ambiental (ou "bruta") e institucional. Além disso, propõe-se analisar alguns trabalhos na área de SMA que também trabalham essa interação, inspirados ou não na teoria de Searle. Ao fim dessa análise, espera-se reunir uma série de elementos que formarão o embasamento teórico para o estudo realizado no restante do trabalho.

2.1 A REALIDADE SOCIAL

A argumentação exposta em (SEARLE, 1995) parte da ideia de que a realidade é composta, em parte, por elementos objetivos e, em parte, por elementos subjetivos. Os elementos objetivos da realidade, chamados de *fatos brutos*, que podem ser descritos pela física, química,

matemática, biologia etc, independente de qualquer crença ou opinião. Pode-se citar, como exemplo, o fato de a água ser composta por oxigênio e hidrogênio. Junto com essa porção objetiva da realidade, existe uma *realidade social*, composta por fatos subjetivos que existem somente porque as pessoas acreditam que eles existem. Uma cédula de papel, por exemplo, pode ser descrita segundo diversas propriedades, como sua composição química, forma, massa, cor etc. O fato de essa cédula, eventualmente, ter atrelado a si um valor monetário e ser considerada como dinheiro, no entanto, não pode ser explicado somente em função dessas mesmas propriedades. Esse fato é um *fato institucional*, pois explica-se pela concordância coletiva de que tal pedaço de papel representa dinheiro para um grupo de pessoas. Outros exemplos de elementos pertententes a essa realidade social são a propriedade privada, o casamento, eleições, governos, partidas de futebol, constituições nacionais etc.

Essa realidade social se origina a partir da realidade bruta: um mundo feito por partículas de matérias dá origem a um mundo feito de conceitos que só existem na mente das pessoas. Ou seja: a realidade social é construída a partir dos elementos físicos que compõem o mundo, da ação das pessoas sobre esses elementos e da interação das pessoas entre si.

2.1.1 Elementos necessários à realidade social

A realidade social está amparada em três elementos essenciais: intencionalidade coletiva, atribuição de função e *constitutive rules*. Esses três elementos serão analisados nas próximas seções. Pelo fato de essa análise ser feita no contexto de SMA, os elementos autônomos que fazem parte da realidade serão tratados por *agentes* e não mais por *pessoas*.

2.1.1.1 Intencionalidade coletiva

A intencionalidade coletiva é o fenômeno pelo qual os agentes compartilham crenças, desejos e intenções e, como consequência, esse compartilhamento afeta a intencionalidade individual de cada agente. Como exemplo, pode-se citar um jogador de futebol desempenhando a função de atacante. Este jogador compartilha crenças, desejos e intenções com o restante da equipe e as crenças, desejos e intenções par-

ticulares do jogador são afetadas por esse compartilhamento: a equipe tem o desejo de vencer uma partida e o agente atacante também tem esse desejo. Em decorrência disso, o agente pode ter desejos particulares como posicionar-se próximo ao gol adversário, adquirir a posse da bola etc.

Fatos que envolvem intencionalidade coletiva são fatos sociais (*social facts*). Dessa forma pode-se definir fatos sociais, que compõem a realidade social, como *fatos em que dois ou mais agentes compartilham crenças, desejos e intenções e que determinam crenças, desejos e intenções particulares desses agentes.* Suponha-se o fato de duas pessoas estarem caminhando lado a lado em uma calçada. Suponha-se, adicionalmente, que essas pessoas estejam caminhando lado a lado devido a uma série de coincidências: chegaram àquele ponto no mesmo momento, vão ao mesmo destino etc. Esse fenômeno não pode ser classificado como um fato social. Se essas pessoas, ao contrário, estivessem caminhando juntas por terem combinado fazê-lo, sejam quais forem as razões para tal, isso seria um *social fact* pois elas estariam compartilhando crenças, desejos e intenções.

2.1.1.2 Atribuição de função

A atribuição de função é o fenômeno pelo qual os agentes atribuem (ou impõem) funções às porções de matéria que compõem o mundo em virtude de sua estrutura física. Uma porção de madeira, disposta da maneira correta, pode ter, para um determinado grupo de agentes, a função de *mesa*. Uma outra porção de madeira, com algumas propriedades diferentes, pode ter a função de *cadeira*. Essas funções, apesar de serem atribuídas em função das propriedades físicas das porções de matéria, só existem pela vontade dos agentes. Sem esses agentes, tais porções de matéria, independente de suas propriedades, seriam apenas porções de madeira.

Uma categoria especial de atribuição de função são as *nonagentive functions* que são funções que ocorrem normalmente mas que são "reconhecidas" pelos agentes. Um exemplo de uma função dessa categoria é a função que o coração tem de bombear o sangue.

2.1.1.3 Fatos Institucionais - *Constitutive Rules*

Conforme mencionado anteriormente, fatos sociais são aqueles em que há um compartilhamento de crenças, desejos e intenções entre dois ou mais agentes. Alguns fatos sociais podem ser classificados como *fatos institucionais*. Além de terem as características que definem os fatos sociais, estes fatos acontecem no contexto de *instituições*.

Para compreender *constitutive rules* e instituições, é necessário introduzir o conceito de *regulative rules*. Uma *regulative rule* é uma regra que regula uma atividade que pode existir independentemente de tal regra. A regra de dirigir um carro pelo lado direito de uma via regula a atividade de dirigir um carro; esta atividade, porém, pode existir independentemente da regra.

As *constitutive rules* não apenas regulam mas, mais do que isso, *constituem* determinadas atividades. As regras de um jogo de xadrez, por exemplo, fazem com que esse jogo exista de fato. Se não houvessem tais regras, duas pessoas poderiam movimentar peças sobre um tabuleiro mas isso não seria um jogo de xadrez.

As *constitutive rules* têm a forma X *count as* Y *in* C, onde X é um fato bruto, Y é um fato institucional e C é o contexto em que o fato bruto X tem como consequência o fato institucional Y. Como exemplo, pode-se citar a instituição de um casamento. O ato de declarar que duas pessoas estão casadas (X), se executado por um sacerdote reconhecido, em cerimônia adequada (C), é socialmente aceito como um ato que confere aos noivos o status de casal (Y). Se, no entanto, o contexto, composto por elementos como sacerdote, cerimônia e outros, não for adequado ao que é aceito coletivamente, os noivos não serão considerados social ou legalmente casados. Conforme Searle, um fato institucional pode gerar outro fato institucional. Por exemplo, um pedaço de papel com determinadas características pode significar, em uma instituição, uma escritura que indica a posse de um imóvel. O fato institucional de o papel ser uma escritura que indica a posse de um imóvel pode ter, por sua vez, o significado institucional de que seu detentor seja o proprietário de tal imóvel.

É possível haver alguma confusão entre os conceitos de *atribuição de função* e *constitutive rules*. Retomando o exemplo utilizado na seção 2.1.1.2, suponha-se que um grupo de pessoas atribua, a um objeto com algumas características específicas, a função de "cadeira". Se em algum momento, apenas uma pessoa do grupo continuar a considerar o objeto como uma cadeira, para essa pessoa, esse objeto continuará tendo tal função, independente da opinião dos demais, pois as características do

objeto permitem que ele continue sendo usado como cadeira. O mesmo não acontece com uma cédula de dinheiro: se apenas uma pessoa, em um grupo, considerar uma cédula como algo que tem valor monetário, essa pessoa não poderá usar a cédula como dinheiro pois as demais pessoas não a aceitarão em troca de algum bem ou serviço. Sendo assim, verifica-se que as *constitutive rules* denotam *imposição coletiva de função*(*status-function*): a função atribuída ao termo X da regra $count - as$ tem significado institucional pois um grupo de agentes concorda com tal atribuição, contida no termo Y (ou aceita a atribuição, no caso de imposição de função).

Em uma regra *count-as*, os termos X e Y podem ser pessoas (ou agentes), objetos ou eventos. No caso de pessoas, pode-se citar o caso de profissionais que, atendendo determinados requisitos, *count as* professores, advogados, sacerdotes religiosos etc. Quanto a eventos, pode-se citar como exemplo eleições e cerimônias de casamento. Com relação a objetos, pode-se citar como exemplo dinheiro, contratos sociais e certificados de propriedade privada.

A atribuição de funções institucionais a objetos e pessoas tem como função principal viabilizar atividades (ou eventos) institucionais. Objetos com função institucional são chamados "objetos institucionais" (*institutional objects*). Embora Searle não mencione ideia semelhante, é possível concluir que, com relação a pessoas, pode-se definir algo como "papeis institucionais". Como exemplos de objetos institucionais pode-se citar novamente dinheiro, contratos sociais, universidades e governos. Como exemplo de papeis institucionais, pode-se citar professores, advogados e senadores.

O processo de criação de fatos institucionais, em que grupos de pessoas concordam sobre funções atribuídas a objetos, pode acontecer sem que os agentes envolvidos na instituição estejam plenamente conscientes a respeito. Nas sociedades humanas, por exemplo, muitos fatos institucionais são incorporados à cultura das pessoas. Como consequência disso, alguns conceitos, principalmente aqueles relacionados a objetos e papeis institucionais, tornam-se usuais às pessoas a ponto de confundirem-se com conceitos brutos. Cita-se novamente o exemplo do dinheiro: nas sociedades atuais, o dinheiro é usado sem que as pessoas reflitam sobre a concordância coletiva a respeito do seu valor ou sobre o fato de os valores monetários terem um valor correspondente em ouro.

2.1.2 Da possibilidade de realidade social em SMA

Com relação aos três elementos componentes da realidade social, pode-se traçar um paralelo entre a teoria de Searle e aspectos relacionados a SMA. Quanto à intencionalidade coletiva, a própria razão de ser de tais sistemas é a possibilidade de dois ou mais agentes atuarem em conjunto de diversas maneiras, como através da cooperação ou competição (WOOLDRIDGE, 1997). Dessa maneira, as crenças, desejos e intenções individuais dos agentes são afetados por crenças, desejos e intenções coletivas, ocasionando, assim, fatos sociais em SMA.

Sobre a atribuição de função em SMA, para interagir com elementos do ambiente, um agente precisa reconhecer em tal objeto alguma função que lhe seja útil. A forma como os agentes interpretam o ambiente e atribuem funções a seus elementos pode variar. Em algumas situações, os agentes podem identificar determinadas propriedades de tais elementos, e, em função disso, raciocinar e proceder a atribuição de função. Em outras situações pode acontecer o que Searle chamou *imposição de função*, que é uma forma especial de atribuição de função. Pela imposição de função, os agentes não precisam atribuir funções aos elementos do ambiente; ao contrário, as funções desses elementos são estabelecidas por elementos externos aos agentes (pelo projetista do sistema, por exemplo) e estes apenas reconhecem e aceitam tais funções. A análise dos aspectos relacionados à atribuição e imposição de função está fora do escopo desse trabalho. Para abstrair aspectos relacionados ao raciocínio dos agentes para efetivar a atribuição de função, utilizar-se-á, neste trabalho, a ideia de imposição de função.

Quanto às *constitutive rules*, alguns elementos em SMA podem ser consideradas como fatos brutos. Partindo-se dessa ideia, pode-se estabelecer mecanismos para dar significado institucional a tais fatos brutos. Alguns autores destacam as interações entre os agentes, principalmente através de atos de fala, como fatos brutos que podem ter significado institucional (JONES; SERGOT, 1996; ARTIKIS; PITT; SERGOT, 2002; ARTIKIS; SERGOT; PITT, 2009; ALDEWERELD et al., 2010, 2009; CAMPOS et al., 2009; ESTEVA et al., 2001). Um exemplo dessa abordagem seria o ato de um agente anunciar um lance em um leilão: nas circunstâncias corretas, o ato de fala do agente teria o significado institucional (*count as*) um lance de leilão. Outros autores destacam as ações dos agentes sobre elementos do ambiente como origem de fatos brutos (PIUNTI, 2009; PIUNTI et al., 2010). Nesse caso, um exemplo seria o fato de um agente passar um semáforo fechado: tal ação teria os significado institucional de uma infração. Esse mesmo exemplo

do semáforo pode ser utilizado para exemplificar uma outra abordagem para *constitutive rules*: determinados fatos institucionais podem originar novos fatos institucionais. No caso de um semáforo, pode-se considerar que o fato de um agente cruzar por um semáforo já é, por si só, um fato institucional. Explica-se isso pois o agente pode cruzar, em seu trajeto, por diferentes tipos de objetos (árvores, postes, outros agentes etc) sem que isso tenha qualquer significado institucional. O fato de o agente cruzar por um objeto "semáforo" e esse semáforo ser considerado um elemento utilizado no controle de tráfego pode ser considerado um fato institucional pois se não houvesse uma instituição que suportasse tal fato, o ato não teria qualquer significado. Em nossa sociedade, o fato institucional de cruzar um semáforo fechado tem significado institucional de uma infração. O estado dos elementos do sistema também pode ter significado institucional. Suponha-se, por exemplo, um cenário que simule uma sala de aula. Suponha-se também que, neste cenário, definiu-se que o fato de um agente estar na sala, independente de como tenha entrado, faça com que este seja considerado um aluno. Nesse caso, estar na sala de aula *count as* assumir o papel de aluno. Uma última forma de transformação de fatos brutos em fatos institucionais é a criação de conceitos institucionais a respeito de elementos existentes na realidade bruta, definindo o que, em (ALDEWERELD et al., 2010), foi chamado "conceito abstrato" (*abstract concept*). Um exemplo para esse caso seria o fato de uma cédula de papel, em determinado contexto, ser considerada como uma cédula com valor monetário. O fato institucional de a cédula ser considerada "dinheiro" não é resultado de interação dos agentes, da atuação dos agentes sobre o ambiente ou do estado do sistema. Em lugar disso, a simples aceitação coletiva faz com que, dentro de uma instituição, aquele elemento tenha uma função institucional específica.

Verifica-se, assim, que os elementos essenciais à construção da realidade social, em que instituições são formadas a partir de *constitutive rules* que transformam fatos brutos em fatos institucionais, estão presentes em SMA. Por esse motivo, à semelhança de outros assuntos relacionados à SMA, as ciências sociais podem apontar um caminho para tratar da interação entre ambiente e instituição. Alguns autores têm tratado desse assunto de diversas maneiras. Algumas das propostas serão analisadas nas próximas seções, buscando assim, formar um embasamento teórico com elementos provenientes das ciências sociais e também da área de SMA.

2.2 A REALIDADE SOCIAL EM SMA

Sistemas multiagentes podem ser considerados como *sociedades de agentes* (COSTA, 2011) devido a características de tais sistemas que são análogas a algumas características verificadas em sociedades humanas (CASTELFRANCHI, 2000; SINGH, 2000; COSTA, 2011; ARTIKIS; PITT; SERGOT, 2002). Grande parte desses autores reconhece que os fatos brutos em SMA (interação entre agentes, ação de agentes no ambiente, eventos ocorridos no ambiente etc) podem ter significado em nível institucional (que pode ter outros nomes como "organizacional", "social" ou "normativo). Em função disso, a ideia de Searle a respeito da criação de fatos institucionais através de *constitutive rules* é reconhecida como uma abordagem aplicável em SMA.

Para verificar a aplicabilidade dos conceitos de Searle em SMA, nesta seção serão analisados algumas propostas feitas por diversos autores. Antes dessa análise, no entanto, serão feitas algumas considerações a respeito de conceitos como "instituição", "organização", "sociedade"em SMA.

2.2.1 Instituições em SMA

Searle aponta para a existência de uma realidade abstrata e subjetiva: a realidade social. Essa realidade é composta por fatos sociais e fatos institucionais. Conforme considerações anteriores, existem elementos em SMA que indicam a possibilidade de que esses sistemas contenham fatos sociais e institucionais, sendo que, neste trabalho, o foco são os fatos institucionais. Para tratar de fatos institucionais, no entanto, é necessário definir claramente o que entende-se por *instituição* neste trabalho.

Da área da economia, em (ALSTON, 1992), instituições são definidas como regras projetadas para guiar as interações humanas, limitando as opções de escolha para as ações das pessoas, influenciando organizações e definindo normas. Segundo o autor, *organizações* são estruturas existentes dentro das instituições, reunindo agentes em torno de um objetivo comum. As organizações buscam, através de sua estrutura, potencializar características dos agentes para atingir seus objetivos dentro dos limites definidos pela instituição. Esses limites são normas que definem obrigações, permissões e proibições dentro da instituição. Seguindo essa ideia, instituições orientam a evolução, fornecem suporte e conferem legitimidade às organizações (ESTEVA et al., 2001).

Em (COSTA, 2011) tem-se a ideia de que instituições são os elementos que compõem uma sociedade de agentes. Segundo o autor, as instituições possibilitam que a sociedade em que estão inseridas mantenha, ao longo do tempo, a coordenação de suas atividades. Uma sociedade, dessa forma, pode ser formada por várias *instituições*, responsáveis pelo desempenho de diferentes funções necessárias ao funcionamento e à persistência da sociedade. Uma instituição, segundo essa abordagem, seria composta por (i) unidades organizacionais, (ii) objetivos, (iii) valores, (iv) normas, (v) atividades funcionais e (vi) funções sociais.

Diversos outros modelos e teorias na área de SMA mencionam estruturas que têm por finalidade guiar e limitar a atuação dos agentes para garantir a integridade do sistema. Essas estruturas podem ser definidas como "instituições", "organizações" e "normas" entre outros. Apesar dessa diversidade, pode-se verificar uma convergência na ideia de que um SMA é uma *sociedade de agentes* que pode ser composta por uma ou mais *instituições*; cada instituição é formada por diferentes estruturas, tais como *organizações* e *normas* e tem como objetivo orientar e restringir a atuação autônoma dos agentes em um SMA, buscando mantê-lo em um estado consistente com a finalidade para o qual foi projetado.

Utilizando-se o termo "instituição" segundo o estabelecido anteriormente, é possível verificar que existem algumas pesquisas que tratam interação entre as dimensões ambiental e institucional em SMA. Nas seções que seguem, serão feitas algumas considerações a respeito dessas pesquisas.

2.2.2 A especificação de Jones e Sergot

Em (JONES; SERGOT, 1996) são feitas algumas considerações a respeito da possibilidade de os atos dos agentes, em determinadas circunstâncias, terem significado institucional. Os atos considerados pelos autores são interações entre os agentes, sem considerar explicitamente a possibilidade de esses atos serem executados sobre algum elemento inserido no ambiente. Não é especificado também *quem é* a instituição que dá significado a um fato bruto. Segundo os autores, um ato de um agente x pode ter significado institucional para um agente y. Este agente y pode ser o próprio agente x, pode ser um outro agente ou

pode ser a instituição[1]. Os autores também ressaltam que além dos atos, o estado de um agente pode ter significado normativo. Uma pessoa com idade superior a um determinado número, por exemplo, pode ser socialmente considerada um idoso.

Além da "conexão" entre alguns atos/estados dos agentes (fatos brutos) e suas consequências em nível institucional, um segundo aspecto tratado pelos autores refere-se às condições em que um ato de um agente pode ter tais consequências. Um ato de um agente terá significado institucional se o agente possuir (i) poder institucional para que o ato tenha significado institucional; (ii) capacidade "física" de levar a termo o ato e (iii) permissão para executar o ato.

2.2.3 A especificação de Artikis *et al.*

Em (ARTIKIS; PITT; SERGOT, 2002; ARTIKIS; SERGOT; PITT, 2009; ARTIKIS; SERGOT, 2010) são descritos sucessivos refinamentos de um modelo que tem forte inspiração no trabalho de (JONES; SERGOT, 1996), utilizando diferentes formalismos e apresentando algumas implementações. A abordagem define que uma sociedade de agentes pode ser expressa como uma composição de:

- Um conjunto de agentes;
- Um conjunto de regras sociais (*social constraints*);
- Um conjunto de papeis a serem desempenhados pelos agentes (*social roles*);
- Estado dos agentes e do ambiente (*social states*);
- Uma linguagem de comunicação;
- Relacionamentos entre os agentes (ex.: propriedade, representação etc);
- A estrutura da sociedade;

Entre os itens citados acima, o modelo proposto foca-se na especificação das regras sociais, papeis e estados sociais.

As regras sociais são definidas em quatro dimensões: (i) capacidade física, (ii) poder institucional, (iii) permissões, proibições e

[1] Os autores não fazem considerações a respeito de como essa instituição é composta.

obrigações e (iv) políticas de punição (*sanctions*) e coação (*enforcement*) para tratar de ações em desacordo com as normas. Observa-se que as três primeiras dimensões são inspiradas em (JONES; SERGOT, 1996).

A *capacidade física* trata da possibilidade (ou da ausência de restrições) para execução de uma ação por parte do agente com relação a fatores externos ao próprio agente e à instituição. Em um ambiente que implemente um jogo de futebol, por exemplo, o agente precisa estar posicionado dentro do campo de jogo para poder chutar a bola. Como outro exemplo, pode-se citar um agente que queira fazer uma compra: para tal, pode-se definir que o agente precise ter o valor correspondente à compra disponível em alguma instituição financeira.

O *poder institucional* (*empowerment*) refere-se a uma característica especial verificada nas instituições: os atos de determinados agentes, cumprindo determinados papeis, podem ter algum significado em nível institucional. Cita-se como exemplo a realização de um casamento: caso o celebrante tenha poder institucional, o ato será efetivamente considerado um casamento; do contrário, não terá significado algum. Uma ação desempenhada por uma agente devidamente habilitado (*empowered*) é denominada uma *ação válida*. A definição de que um agente tem, em dado momento, poder institucional, pode ser dada em função de diversos fatores. Suponha-se uma situação em que um agente a apresente uma proposta de compra de um produto a um agente b. Se o agente a estiver habilitado, seu ato significará, efetivamente, uma proposta de compra; do contrário, não terá significado algum. São exemplo de situações em que a está habilitado:

- a ocupa um papel apropriado: a ocupa o papel de comprador;
- a e b ocupam papeis apropriados: a ocupa o papel de comprador e b ocupa o papel de vendedor;
- a e b ocupam papeis apropriados e o ato realizado por a é feito em momento apropriado: a ocupa o papel de comprador e b ocupa o papel de vendedor e b fez uma proposta de venda a a;
- a e b ocupam papeis apropriados e o ato realizado por a é feito da maneira correta: a ocupa o papel de comprador e b ocupa o papel de vendedor e a oferta feita por a corresponde ao valor pretendido por b.

É importante mencionar que os exemplos são baseados em situações hipotéticas com a finalidade de exemplificar alguns dos fatores

que podem, no modelo descrito, definir o poder institucional de um agente.

As *permissões*, *proibições* e *obrigações* são um outro nível na especificação das regras sociais. Uma ação permitida pode, por vezes, ser confundida com uma ação válida. Os autores, no entanto, enfatizam uma distinção entre os dois níveis: um agente pode estar habilitado para criar um fato institucional e, ao mesmo tempo, devido a circunstâncias momentâneas, estar proibido para tal. Utilizando novamente o exemplo de uma relação de compra e venda entre dois agentes, um agente pode estar "fisicamente capacitado" a fazer uma compra (possuindo recursos financeiros para tal) e pode estar habilitado (desempenhando o papel de comprador), mas, ainda assim pode, circunstancialmente, estar proibido de executar tal ação por ter excedido o limite de itens permitidos a um único comprador.

Os mecanismos de *sanção* e *coação* são os meios a serem utilizados para inibir comportamentos indesejados por parte dos agentes. Quanto às sanções, o modelo preocupa-se com dois aspectos: (i) definir quais situações são passíveis de sanções e (ii) definir qual o tipo de sanção a ser aplicada. Quanto à coação, o modelo prevê tal mecanismo mas não lhe dá enfoque por considerar que sua utilização é um limitador à liberdade dos agentes.

Além das regras sociais, um outro componente de uma sociedade de agentes é o conjunto de papeis que compõem essa sociedade. Segundo os autores, os agentes membros de uma sociedade devem ter no mínimo um papel. Para adotar um papel o agente deve cumprir determinados requisitos (específicos da aplicação e do papel); ao adotar um papel, o agente passa a ter parte de seu comportamento determinado pelo papel ocupado.

Para descrever os conceitos, foram utilizados dois formalismos diferentes: a linguagem $C+$(ARTIKIS; SERGOT; PITT, 2009) e o *Event Calculus* (ARTIKIS; PITT; SERGOT, 2002; ARTIKIS; SERGOT, 2010). Através desses formalismos, são definidos estados institucionais que uma ação pode iniciar ou terminar. Com base nesses formalismos, foram implementadas duas ferramentas que armazenam o estado da sociedade (em termos de capacidades físicas, poder institucional, permissões/proibições/obrigações e papeis) e, em função das sucessivas ações dos agentes, deduzem a evolução de tal estado.

À semelhança do modelo apresentado em (JONES; SERGOT, 1996), os modelos descritos nesta seção consideram que os atos são provenientes das interações entre os agentes. Verifica-se, nesses modelos, no entanto, uma estrutura, externa aos agentes, que armazena a

especificação da "instituição" e que, em tempo de execução, é capaz de processar sua evolução em função dos atos dos agentes.

2.2.4 MASQ

O modelo MASQ é uma evolução dos modelos AGRE (FERBER; MICHEL; BÁEZ-BARRANCO, 2004) e AGREEN (BÁEZ-BARRANCO; STRATULAT; FERBER, 2007) e parte do princípio de que as interações entre os agentes acontecem através do ambiente (o "ambiente", nesse caso, refere-se a tudo o que é externo aos agentes). Esse ambiente pode ser particionado em *spaces* que abrigam porções do ambiente, divididas em duas categorias: física (*physical space*) e social (*social space*). Como exemplo, podemos citar um cenário envolvendo uma partida de futebol. O *space* físico seria a parte do mundo que envolve elementos como campo de futebol, bola e uniformes das equipes. O *space* social seria o conjunto de regras necessárias à realização da partida, como, por exemplo, definição das equipes, esquemas táticos, regras da partida e fórmula do campeonato.

Os aspectos físicos e sociais do sistema, no modelo AGRE são colocados em um mesmo nível de abstração, ambos sendo considerados aspectos pertencentes ao ambiente, pois, segundo os autores, as características fundamentais destes elementos são semelhantes: o estado dos *spaces* independe da percepção dos agentes e modifica-se mediante a ação dos agentes. Além disso, os elementos que compõem o *spaces* tem natureza reativa e determinista.

No modelo AGREEN o conceito de *space* é explorado e refinado, de forma a possibilitar a modelagem de influências entre diferentes *spaces* através de ligações (*links*). Estas ligações podem ser: (i) Causais: definem a ocorrência de eventos locais em função de eventos ocorridos em outro *space*; (ii) Lógicas: definem a alteração de propriedades locais em função de propriedades verificadas em *spaces* externos; (iii) Sociais: relações entre *spaces* físicos e sociais, com significado semelhante às regras *count as*. Os tipos de ligações que um *space* admite definem sua classificação em físico ou social: um *space* é considerado "social" se possuir ligações sociais com outros *spaces*.

Os conceitos desenvolvidos no modelo AGREEN foram utilizados por (STRATULAT; FERBER; TRANIER, 2009) na proposta do modelo MASQ, em que as interações dos agentes com infraestruturas que modelam aspectos ambientais (tanto físicos quanto sociais) se dão através dos *spaces*. Essas interações entre agentes e *spaces* são a origem da

realidade institucional. Essa transformação, no MASQ, acontece da seguinte forma: os agentes percebem os fatos brutos e aplicam, sobre esses fatos, as *constitutive rules* adequadas, gerando fatos institucionais. Estas *constitutive rules* podem ser formais, possuindo alguma forma de representação no *brute space* e sendo acessíveis aos agentes (semelhante a uma lei que é publicada), ou informais, sob a forma de conhecimento e valores compartilhados e aceitos pelos membros da instituição. Essa abordagem faz com que não haja representação da realidade institucional fora da mente dos agentes. Ao contrário, cada agente tem sua própria interpretação da realidade institucional, formada conforme as *constitutive rules* que possui internalizadas e conforme sua percepção particular dos fatos brutos. Como consequência disso, os autores apontam a possibilidade de que cada agente tenha uma interpretação diferente, parcial e até mesmo inconsistente da realidade institucional.

2.2.5 *Situated Electronic Institutions*

As *Eletronic Institutions* (EI), propostas em (ESTEVA, 2003), são um *framework* que aplica conceitos institucionais às interações em SMA. Mais especificamente, o *framework* busca garantir que as ações dialógicas dos agentes sejam realizadas de forma a manter o sistema em um estado consistente. Para prover tal garantia, ações não permitidas são bloqueadas pela EI.

Os agentes, no modelo EI, são considerados elementos externos à instituição e interagem com ela através de interfaces chamadas *governors*. As ações da instituição são executadas por agentes especiais chamados *staff agents*. Um esquema de uma EI é exibido na Figura 2.

As EI foram definidas, orignalmente, como sendo autocontidas e estáticas. Segundo essas características, a instituição deve ser projetada de acordo com o ambiente em que será inserida para que a interação entre ambiente e instituição possa ser prevista em tempo de projeto. Essa característica, no entanto, conflita com a ideia um SMA aberto, em que entidades não conhecidas de antemão (agentes, outras instituições, elementos inseridos no ambiente etc) podem vir a interagir com o sistema. Para tratar dessas limitações, em (CAMPOS et al., 2009) é proposta uma extensão às EI, denominada *Situated Electronic Institutions* (SEI). Uma SEI, nesse caso, seria uma instituição "situada" em um ambiente do qual recebe e sobre o qual exerce influências. Os elementos que compõem esse ambiente (agentes, organizações, outros SMA et) podem ser projetados independemente da instituição.

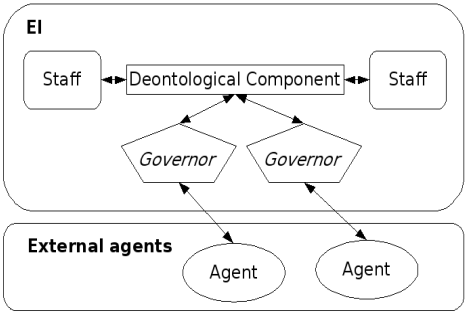


Figura 2 – *Electronic Institution*(CAMPOS et al., 2009)

Uma instituição, no modelo de SEI, armazena três conjuntos elementares de informações:

- Propriedades dos agentes com relação à instituição (*Agent Properties* - P_A): informações de interesse da instituição a respeito dos agentes que interagem com ela. Ex.: papeis desempenhados por um agente em uma organização;
- Propriedades do ambiente que não são influenciadas pela instituição. Ex.: data, temperatura;
- Propriedades da instituição: fatos influenciados, direta ou indiretamente, pela instituição. Ex.: violação de normas.

A interação entre a instituição e o ambiente é feita através de dois tipos de agentes especializados chamados *Modellers* e *Staff Agents*. Um *Modeller* é equivalente a um *Governor* no modelo EI e faz a interface entre a instituição e os agentes. As atividades resultandes do processamento da instituição (como a aplicação de normas, por exemplo) são feitas por *staff agents*. A Figura 3 ilustra esses elementos em um cenário relacionado ao tráfego viário. Verifica-se na figura um *Modeller* que faz a intermediação entre um carro e a instituição (neste caso, o carro é considerado um agente que interage com a instituição). Esse *modeller* capta do ambiente, através de uma câmera, uma P_A que define a posição do veículo. Nesse mesmo cenário, um policial pode solicitar que o *modeller* altere uma propriedade referente à pontuação atribuída ao condutor do veículo em função de eventuais infrações. Nesse caso, o *modeller* irá interagir com elementos apropriados no ambiente para atender à solicitação do policial. Verifica-se, também, na Figura 3, a

existência de dois elementos *staff*, que aplicam decisões da instituição, processadas pelo componente deôntico (*deontological component*), aos elementos do ambiente. Em um dos casos, o *staff signals* controla um semáforo; em outro caso, um *staff police* define se um veículo cometeu alguma infração e, nesse caso, interage com o *modeller* adequado, e esse, por sua vez, interage com o elemento do ambiente que controla as infrações do veículo, conforme descrito anteriormente. Para viabilizar a interação entre uma SEI e o ambiente, o modelo prevê a utilização de interfaces denominadas *bridges*. As *bridges* são apenas um conceito no modelo de SEI mas podem ser definidas como APIs específicas para os diferentes elementos do ambiente que interagem com a instituição. O componente deôntico (*Deontological Component - DC*), citado anteriormente, abriga uma estrutura normativa (*Normative Structure - NS*) que define ilocuções que são proibidas, permitidas ou obrigatórias a agentes que desempenham deteriminados papeis.

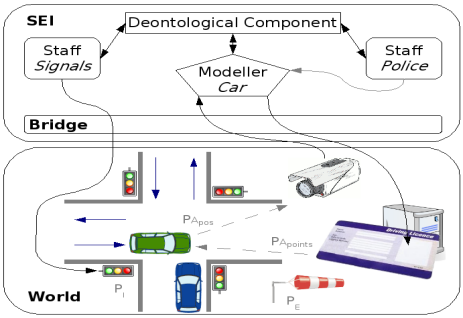


Figura 3 – *Situated Electronic Institution*(CAMPOS et al., 2009)

Os elementos descritos anteriormente (*modellers*, *staff agents*, *bridges* e *deontological component*) fornecem a infraestrutura que viabiliza a interação entre organização (ou instituição) e ambiente. As ações ocorridas no ambiente que influenciam a instituição são chamadas *relevant actions*. As *relevant actions* podem ser classificadas em dois grupos: ações permitidas (*allowed actions*) - que estão de acordo com as normas da instituição - e ações proibidas (*non-allowed actions*) - que não estão de acordo com as normas da instituição, definidas na *NS*.

Verifica-se com essa descrição que as SEI provém suporte criação de fatos institucionais. Os fatos brutos que originam os fatos institucionais no modelo de SEI, no entanto, seriam resultantes, basicamente,

das ilocuções dos agentes, não configurando, dessa forma, uma interação organização-ambiente.

2.2.6 O modelo de Dastani *et al.*

Em (DASTANI; TINNEMEIER; MEYER, 2009) e (DASTANI et al., 2009) é proposta uma linguagem que permite escrever regras que determinem as consequências normativas das ações dos agentes no ambiente. A proposta parte da ideia de que o SMA é um sistema que observa as ações dos agentes, determina mudanças de estado decorrentes de tais ações, verifica se o sistema está em um estado que viola alguma norma e aplica sanções quando necessário. A linguagem proposta descreve (i) regras que determinam a mudança de estado do ambiente em função de ações dos agentes (*Facts*), (ii) regras que determinam o efeito normativo de determinados estados do ambiente (*Count-as rules*), (iii) punições para eventuais violações das normas (*Sanction rules*).

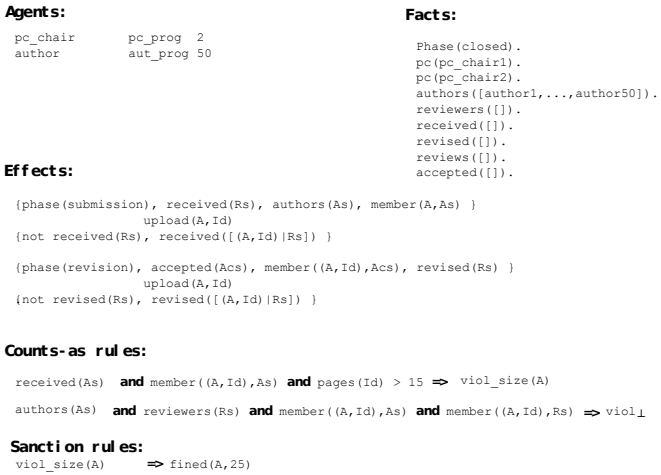


Figura 4 – Exemplo de programa normativo (Adaptado de (DASTANI; TINNEMEIER; MEYER, 2009))

Um exemplo do programa proposto é exibido na Figura 4 e trata de um sistema de submissão de artigos a um evento. A cláusula *Agents* define os agentes que irão compor o sistema. A cláusula *Facts* define os fatos brutos iniciais, determinando assim o estado inicial do sistema. A cláusula *Effects* define, através de sentenças na forma {{*contexto*} *ação*

{*consequência*}}, consequências para as ações dos agentes. Os termos {*contexto*} e {*consequência*} são conjuntos de predicados relacionados ao estado do ambiente e o termo *ação* se refere a ações que podem ser executadas pelos agentes. O primeiro efeito do exemplo exibido na Figura 4 indica que se um agente A fizer *upload* de um artigo Id durante a fase de submissão, o artigo será incorporado ao conjunto dos artigos recebidos.

O modelo define que os estados do ambiente podem ter efeito normativo. Ou seja, o ambiente pode estar em um estado que é considerado "proibido". Esse efeito normativo é definido na seção *Count-as rules* do programa. A primeira regra exibida na Figura 4, por exemplo, determina que se um agente A submeter um artigo Id e esse artigo possuir mais que 15 páginas, essa submissão é uma violação de uma norma do sistema.

A cláusula *Sanctions* determina punições como consequência das violações. A primeira sanção definida no programa determina que um agente que submeteu um artigo com mais de 15 páginas será multado.

Verifica-se, assim, que os autores buscam propor uma linguagem que descreva a evolução do estado do ambiente, as consequências institucionais (ou normativas) de alguns estados e eventuais sanções a violações das normas. A proposta, no entanto, limita-se a definir tal linguagem, sem definir como esta seria interpretada ou incorporada a uma estrutura de software que a processasse. A proposta considera a dimensão institucional como sendo prioritariamente normativa: o estado do ambiente determina violações de normas e eventuais sanções. O estado do ambiente, nesse caso, é verificado em função dos predicados declarados no programa, sem considerar o ambiente como uma abstração primária.

2.2.7 O modelo de Aldewereld *et al.*

Em (ALDEWERELD et al., 2009) e (ALDEWERELD et al., 2010) é proposto um modelo para a criação de fatos institucionais fortemente baseado na teoria de Searle. Os autores identificam a utilização do operador *count-as* de três formas diferentes: (i) classificação, (ii) criação de fatos institucionais sem significado normativo e (iii) criação de fatos institucionais normativos. Para fins de simplificação na argumentação, nesta seção será utilizada a expressão "fatos institucionais" para referir-se a "fatos institucionais sem significado normativo"; para referir-se a fatos "institucionais normativos", será utilizada a expressão "fatos

normativos".

A utilização do operador *count-as* para classificação refere-se à definição de significado institucional a objetos que, fora da instituição, não teriam tal classificação. Uma cédula de dinheiro é um exemplo bastante comum e apropriado: um pedaço de papel tem atribuído a si um valor monetário em virtude de uma instituição e de um contexto.

Com relação à utilização do operador *count-as* para criação de fatos institucionais e criação de fatos normativos, os autores afirmam que, para que haja a criação de um fato normativo (a violação de uma norma, por exemplo), deve ter acontecido, anteriormente, um fato institucional. Para explicar esse ponto, pode-se tomar por base o programa exibido na Figura 4. Na abordagem de (DASTANI; TINNEMEIER; MEYER, 2009) e (DASTANI et al., 2009), o fato de um artigo ter sido recebido para submissão a um congresso é um fato bruto que pode implicar em um fato institucional normativo de violação da norma referente ao número máximo de páginas. Segundo a abordagem descrita nesta seção, no entanto, o recebimento de um artigo para submissão já é um fato institucional, pois sem a instituição que suporta o congresso, não haveria submissão.

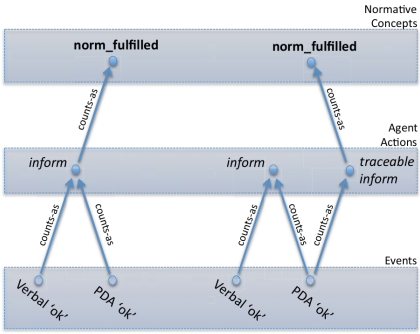


Figura 5 – Exemplo de programa normativo(ALDEWERELD et al., 2010)

Essa ideia é ilustrada na Figura 5, retirada de (ALDEWERELD et al., 2010). Nesse exemplo, os agentes precisam estabelecer comunicações de maneira apropriada para agirem de acordo com as normas do sistema. Em alguns casos, a forma apropriada de comunicação pode ser uma informação (*inform*) qualquer; em outros casos, a comunicação precisa ser feita através de informações rastreáveis (*traceable inform*). Duas formas de comunicação são previstas em tal cenário: comunicação verbal (*verbal ok*) e comunicação através de PDA (*Personal digital as-*

sistant) . Essas duas formas de comunicação são considerdas *inform*, mas somente a comunicação por PDA é consierada *traceable inform*. Sendo assim, em determinados contextos, um *inform* tem significado normativo de norma cumprida. Nesses casos, sendo comunicação verbal e comunicação por PDA duas instâncias de *inform*, estas duas formas de comunicação estão de acordo com as normas do sistema.

Em outras situações, somente *traceable informs* são aceitas pelas normas do sistema. Nesse caso, somente comunicação por PDA seria uma forma de comunicação aceitável. Suponha-se, utilizando-se esse mesmo exemplo, que fosse introduzida no sistema a possibilidade de comunicação por correio eletrônico e que essa comunicação fosse considerada como *traceable inform*, essa forma de comunicação seria automaticamente aceita nas mesmas situações em que a comunicação por PDA é aceita.

A partir destas premissas, os autores citam a implementação de protótipo de um *reasoner* para regras *count-as* sob a forma de um programa DROOLS. Apesar disso, no entanto, não é citada nenhuma aplicação concreta do *reasoner* em um SMA.

2.2.8 *Embodied Organizations*

Em (PIUNTI, 2009; PIUNTI et al., 2010) é exposto um modelo que trata de SMA utilizando três abstrações primárias distintas: agentes, organizações e ambiente. Os agentes constituem as entidades computacionais autônomas, que têm capacidade, raciocinam e tomam decisões. A organização é a infraesturutra que armazena informações sobre normas, papeis, objetivos globais etc, com a finalidade de possibilitar que os agentes tenham comportamento condizente com o esperado pela organização. O ambiente, por sua vez, incorpora entidades que não são modeláveis sob a forma de agentes ou organizações, mas que fornecem suporte aos agentes para que estes busquem atingir seus objetivos. Estas três dimensões interagem entre si, seguindo os seguintes princípios básicos:

- Agentes interagem com ambiente e organização buscando atingir seus objetivos;
- Entidades organizacionais controlam o comportamento dos agentes e regulam a utilização de recursos, procurando manter o estado desejado do ambiente;
- Elementos do ambiente afetam agentes e organização.

O modelo foi desenvolvido considerando a utilização de modelos existentes e implementados para cada uma das dimensões. Mais especificamente, a linguagem *Jason* para programação dos agentes (BORDINI; HÜBNER; WOOLDRIDGE, 2007), $\mathcal{M}$oise para modelagem e implementação da organização (HÜBNER; SICHMAN; BOISSIER, 2007) e CArtAgO para implementação do ambiente (RICCI; PIUNTI; VIROLI, 2011). A partir desta estrutura baseada em três dimensões distintas, surge a noção de *Embodied Organization*, em que a organização é vista como uma parte do SMA cujo estado sofre e exerce influência do ambiente.

As infraestruturas de organização e ambiente seguem o modelo *Agents & Artifacts* (*A&A*) (OMICINI; RICCI; VIROLI, 2008), que tem por base entidades chamadas artefatos. Estes artefatos são entidades reativas que disponibilizam informações e funcionalidades aos agentes para que estes atinjam seus objetivos. Para tanto, são definidas duas infraestruturas distintas, ambas compostas por artefatos: uma infraestrutura organizacional (OMI) e uma infrastrutura ambiental (EMI). Assim, além de os agentes poderem interagir com a organização e com o ambiente, existe a possibilidade de serem definidas interações entre organização e ambiente.

Os artefatos da EMI e OMI estão inseridos em um *workspace*, que pode ser considerado uma partição do ambiente em que encontram-se os recursos, tanto físicos quanto organizacionais, necessários à realização de um determinado conjunto de atividades dos agentes. O conceito de *embodied organization* surge dessa infraestrutura: uma organização pode estar incorporada em uma estrutura mais elaborada em que os fatos que modificam o estado da organização são originados no ambiente e alguns fatos verificados no ambiente são consequência de eventos ocorridos na organização. Essa ideia é ilustrada na Figura 6.

Essa ligação entre as dimensões ambiental e organizacional concretiza aspectos da construção da realidade social proposta por Searle: eventos ocorridos no *workspace* transformam-se em fatos institucionais através de *constitutive rules*.

2.2.9 Análise dos modelos

Os modelos descritos anteriormente tratam, de diferentes maneiras, das consequências institucionais de fatos pretencentes à realidade "bruta"do sistema (interações entre os agentes, atuação dos agentes no ambiente, estado do sistema etc). Através da análise realizada, foi

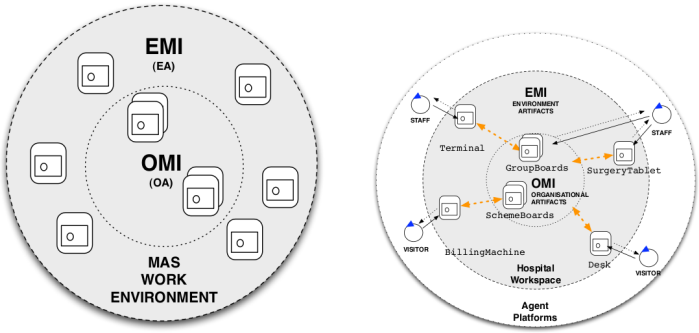


Figura 6 – *Embodied Organization* (PIUNTI et al., 2010)

possível identificar uma série de aspectos, em nível conceitual, que são levados em conta na elaboração dos modelos estudados. Estes aspectos são resumidos na Tabela 1 e serão apresentados a seguir.

2.2.9.1 Abstrações primárias

Por abstração primária (*first-class abstraction*) entende-se uma abstração que não depende de outras formas de abstração para ser representada. Os agentes, por exemplo, são abstração primária em todos os modelos analisados pois, em um SMA, podem até interagir e explorar outras formas de abstração, mas, apesar disso, podem ser modelados sem depender de conceitos relacionados a outras formas de abstração.

Quanto à dimensão institucional, à exceção do modelo MASQ, todos os demais consideram-na como abstração primária. Conforme descrito anteriormente, no MASQ, a dimensão institucional é resultado da interpretação particular dos fatos brutos por cada agente. Sendo assim, essa dimensão depende da abstração "agente" para existir. Nos demais modelos, embora seja expressa de formas diferentes (papeis,

Tabela 1 – Comparação entre modelos

Modelo	Abstrações Primárias			Fatos Brutos					Linguagem			Interação Inst-Amb
	Ag	Amb	Inst	Int	Amb	Est	Inst	Conc	Ling	Interp	Apl	
Artikis *et al.*	✓		✓	✓					✓	✓	✓	
MASQ	✓	✓			✓							
SEI	✓		✓	✓								
Dastani *et al.*	✓		✓			✓			✓			
Aldewereld *et al.*	✓		✓	✓			✓	✓	✓	✓	✓	
Emb.Org.	✓	✓	✓		✓				✓			✓

grupos, normas, sanções), a dimensão institucional pode ser modelada de forma independente de outras formas de abstração.

O ambiente é considerado abstração primária nos modelo *Embodied Organization* e MASQ. Nos outros modelos, o ambiente é considerado como elemento externo ao sistema ou então alguns de seus aspectos são integrados à modelagem da instituição (como no caso do modelo de Dastani *et al.*). Apesar dessa semelhança, os modelos *Embodied Organization* e MASQ são diferentes pois o MASQ considera que o SMA é composto pelas dimensões *agentes* e *ambiente* enquanto o modelo *Embodied Organization* considera que o SMA é composto pelas dimensões *agentes*, *ambiente* e *instituição*.

2.2.9.2 Fatos brutos

Os diferentes modelos apresentam diferentes formas de capturar os fatos brutos que podem vir a ter consequências institucionais. Verifica-se que a maioria dos modelos considera as interações dos agentes, especialmente através de atos de fala, como fatos brutos que podem vir a ter consequências em nível institucional. A atuação de agentes sobre elementos do ambiente também pode produzir fatos institucionais. Cita-se como exemplo, novamente, a situação em que um agente cruza um semáforo fechado: a atuação do agente sobre o objeto (semáforo) significa uma infração. Entre os modelos analisados, somente o modelo *Embodied Organizations* trata da criação de fatos institucionais segundo essa abordagem. Esse mesmo modelo também é o único a tratar o ambiente como uma abstração primária e é possível estabelecer-se uma relação entre essas duas características, uma vez que o elemento pertencente ao ambiente é parte fundamental na produção do fato institucional.

O modelo de Dastani *et al.* considera o estado do sistema como fato bruto gerador de fatos institucionais. Percebe-se, nessa proposta, que, em um primeiro momento, são definidas as consequências das ações dos agentes. Essas consequências são alterações no estado do sistema. As regras *count-as*, por sua vez, são modeladas em função dos estados verificados no ambiente. Ou seja: ao definir regras *count-as*, nessa abordagem, não é necessário levar em consideração as ações que conduziram o sistema a um determinado estado. Ao contrário, é necessário apenas definir consequências institucionais para os possíveis estados a que o sistema possa ser conduzido.

Os fatos institucionais também podem gerar novos fatos insti-

tucionais. Cita-se como exemplo, novamente, a situação descrita na seção 2.2.7, em que, em certas situações, uma comunicação do tipo *inform* significa, institucionalmente, uma violação às normas do sistema. O modelo de Aldewereld *et al.* trata dessa possibilidade ao conferir consequências normativas aos fatos institucionais.

Uma última forma de fato bruto que pode ser convertido em fato institucional são os conceitos dos objetos. Cita-se novamente o exemplo da seção 2.2.7: uma mensagem por PDA, em nível institucional, é considerada uma informação rastreável. Essa ideia é semelhante à ideia, também descrita anteriormente, da cédula de dinheiro: um objeto existente na realidade física (cédula de papel) tem um significado específico em nível institucional (cédula de dinheiro).

2.2.9.3 Linguagem

A forma de representar interações entre ambiente e instituição é um ponto importante a ser considerado na análise realizada. Agentes são representados em termos de crenças, desejos, intenções, objetivos etc; instituições são representadas através de grupos, papeis, normas etc; ambientes são representados através de artefatos, *spaces* etc. A interação entre instituição e ambiente, no entanto, precisa "conectar" elementos das duas dimensões. Alguns modelos furtam-se de descrever uma linguagem para tal (como é o caso do MASQ e SEI). Os modelos de Dastani *et al.* e *Embodied Organization* descrevem uma linguagem, porém não mencionam implementação de interpretador para a linguagem e nem exemplos de utilização da linguagem em um SMA. Os modelos de Artikis *et al.* e Aldewereld *et al.*, por sua vez, descrevem uma linguagem, mencionam a existência de um interpretador e citam sua utilização em SMA, ainda que sejam meros exemplos sem aplicação real.

2.2.9.4 Interação Instituição-Ambiente

O modelo *Embodied Organization* prevê a interação Instituição-Ambiente, em que eventos ocorridos na instituição têm algum impacto no ambiente. Os demais modelos não prevêem, explicitamente, esse tipo de interação.

2.3 CONCLUSÕES

Nas sociedades humanas, as instituições evoluem a partir de fatos verificados em nível "bruto". Essas instituições são formadas por regras que definem como fatos pertencentes a esse nível bruto se transformam em fatos institucionais. Essas regras são chamadas *constitutive rules*.

Os SMA apresentam algumas características que indicam a possibilidade de interação entre ambiente e instituição à semelhança do que acontece em sociedades humanas. Esas características são intencionalidade dos agentes, intencionaliade coletiva e atribuição (ou imposição) de função aos objetos existentes no ambiente.

Ao lado desse paralelo que indica tal semelhança, estudos em SMA mostram diferentes abordagens à interação entre ambiente e instituição. Algumas dessas abordagens foram analisadas e, sob a ótica do presente estudo, algumas características dos diferentes modelos mostraram-se importantes e merecem considerações. Verificou-se que apenas o modelo de (PIUNTI, 2009) considera o ambiente como abstração primária e isso tem importante relação com a forma como o modelo é desenvolvido. Verificou-se também que os diferentes modelos consideram diferentes formas de fatos brutos, identificando-se cinco diferentes formas de fatos brutos que podem ter consequências em nível institucional. Um outro aspecto a ser considerado é a representação da interação ambiente-instituição: alguns modelos não prevêem linguagem para representar tal interação; outros modelos propõem linguagem mas, entre esses, nem todos propõem interpretador para a linguagem e nem todos citam exemplos de aplicação da linguagem e interpretador em um SMA.

Uma outra conclusão decorrente do estudo realizado foi com relação à influência da instituição sobre a própria instituição: um fato institucional pode ser considerado um fato bruto em uma *constitutive rule*. Isso indica a possibilidade de tratar de fatos verificados na dimensão institucional de um SMA como fatos capazes de provocarem alterações na instituição. Essa abordagem é verificada na teoria de Serale e no modelo de (ALDEWERELD et al., 2009) e será adotada no restante deste trabalho.

Conforme mencionado anteriormente, a única proposta em que um SMA é concebido da mesma forma como é concebido neste trabalho (composto por agentes, ambiente e instituição como dimensões primárias) é o modelo de (PIUNTI, 2009). Este modelo, no entanto (i) é proposto com forte embasamento em modelos específicos de ambiente e instituição, (ii) propõe uma linguagem mas não descreve interpretador

para a linguagem proposta e (iii) considera apenas eventos ocorridos no ambiente como fatos brutos. Com base nesses aspectos, no próximo capítulo será proposto um modelo para tratar da mudança de estado da instituição em função de fatos ocorridos no ambiente ou na própria instituição em SMA[2]. O modelo tem inspiração naquele proposto em (PIUNTI, 2009), mas (i) é generico quanto a modelos de ambiente e instituição, (ii) considera tanto eventos quanto estados como fatos brutos e (iii) tem uma linguagem proposta e, para tal linguagem, tem proposto um interpretador.

[2] A "mudança de estado da instituição" será, algumas vezes, referida como *fato institucional*, à semelhança da terminologia usada por John Searle.

3 UM MODELO PARA INTERAÇÃO INSTITUIÇÃO-AMBIENTE EM SMA

No capítulo anterior, foi analisada a relação entre a realidade bruta, perceptível aos sentidos das pessoas, e a realidade institucional. A questão foi analisada tanto sob a ótica das ciências humanas quanto de SMA. Na área das ciências humanas, analisou-se a teoria da construção da realidade social proposta por John Searle (SEARLE, 1995), segundo a qual certos elementos da realidade física têm significado em nível institucional. Esse significado é definido por *constitutive rules* (que, no contexto do modelo proposto, também serão referidas como "regras *count as*"). Os fatos resultantes da aplicação de uma regra *count as* são chamados *fatos institucionais*. De forma semelhante e, em alguns casos, com inspiração na teoria de Searle, diversos estudos tratam de relações entre as dimensões bruta e institucional em SMA e propõem modelos para tratar tais interações.

Neste capítulo é apresentado um modelo para criação de fatos institucionais a partir de fatos ocorridos no ambiente e na instituição, considerando um SMA como um sistema aberto no qual agentes, instituição e ambiente são abstrações primárias distintas. Essa concepção é semelhante à concepção do modelo *Embodied Organization*, apresentado na seção 2.2.8, que inspirou o modelo aqui apresentado. Apesar dessa influência, no entanto, o modelo apresentado neste capítulo apresenta algumas diferenças em relação ao que o inspirou. Uma primeira diferença é que o modelo *Embodied Organization* foi proposto utilizando como base o modelo CArtAgO para infraeastrutura ambiental e o modelo $\mathcal{M}$oise para estrutura institucional. De forma diferente, o modelo aqui apresentado foi elaborado procurando manter generalidade suficiente para abstrair aspectos relacionados a modelos específicos de ambiente e instituição. Essa busca por generalidade implica em outras diferenças com relação ao modelo *Embodied Organization*, como, por exemplo, a utilização do estado do ambiente e da instituição como fato bruto e a definição de uma linguagem específica para descrever a criação de fatos institucionais.

Para apresentar o modelo, este capítulo está estruturado da seguinte maneira:

- Na seção 3.1 são feitas considerações iniciais sobre os principais aspectos relacionados ao *contexto* em que o modelo é proposto: SMA como um sistema aberto formado por três abstrações primárias: agentes, instituição e ambiente.

- Na seção 3.2 é descrito o modelo de interação I-E proposto.

3.1 CONCEPÇÃO DE SMA PARA APRESENTAÇÃO DO MODELO

Conforme mencionado anteriormente, neste trabalho será considerado que um SMA é um sistema aberto e composto, essencialmente, por três formas de abstração primárias (ou *dimensões*): agentes, instituição e ambiente. Por abstração primária entende-se uma abstração independente que modela domínios claramente definidos, podendo interagir com outras formas de abstração mas não dependendo de outras formas de abstração para ser modelada (WEYNS; OMICINI; ODELL, 2007).

Um SMA aberto pode ser definido como aquele em que nem a quantidade, nem o comportamento e nem a forma como os agentes interagirão com os recursos do sistema pode ser totalmente conhecida em tempo de projeto (PIUNTI, 2009). Isso abre a possibilidade para problemas como conflitos entre os objetivos dos agentes e os objetivos da sociedade de agentes, confiabilidade duvidosa com relação aos agentes, inconformidade da ação efetiva dos agentes em relação ao especificado inicialmente, entre outros. Para tratar dessas questões em SMA abertos, diversas abordagens apontam para a utilização de mecanismos de caráter institucional, tais como papeis, normas, mecanismos de punição e coação etc, regulando e orientando as atividades dos agentes com a finalidade de garantir a consistência do sistema. Esses mecanismos, na abordagem utilizada neste trabalho, são reunidos em uma *dimensão institucional* que é tratada como uma abstração primária. Isso significa que a dimensão institucional, com os diversos mecanismos que a compõem, é posta "lado a lado" com os agentes em um SMA, não dependendo, assim, funcional ou conceitualmente, dos agentes ou de quaisquer outros elementos. Uma das implicações dessa abordagem está no fato de que a instituição passa a ser um elemento que não existe somente na mente dos agentes, mas, ao contrário, está presente no SMA independente da presença dos agentes, de suas percepções, crenças ou ações. Dessa forma, mesmo que os agentes não percebam, a instituição estará regulando suas atividades; ou ainda, mesmo que os agentes não desejem seguir o que a instituição estabelece, ainda assim, estarão sujeitos a ela. Isso faz com que, em última análise, a instituição seja capaz de se manter consistente independente de aspectos relacionados a agentes e ambiente.

Juntamente com agentes e instituição, outra forma de abstração

primária a ser considerada é o ambiente que, segundo (WOOLDRIDGE, 1997), é o elemento sobre o qual o agente atua e do qual tem percepções que influenciam em sua atuação. Algumas abordagens tratam o ambiente como algo modelado na "mente" dos agentes e, em função disso, uma representação total do ambiente em um SMA seria a síntese de todas as interpretações do ambiente feitas pelos agentes. Essa visão é subjetiva, pois está sujeita às percepções dos agentes, tendo algumas implicações importantes. Se, por exemplo, dois agentes tiverem percepções diferentes sobre um mesmo elemento do ambiente, não é possível estabelecer uma representação única consistente do ambiente. Além disso, se em algum momento um agente sair do sistema, a representação total do ambiente pode se modificar devido às percepções do agente que não estão mais disponíveis. Uma outra forma de conceber o ambiente em SMA é considerá-lo como uma abstração primária, com funções específicas e diferentes das funções dos agentes e da instituição. Alguns exemplos de funções que podem ser atribuídas ao ambiente são infraestrutura de comunicação, modelagem espacial, suporte a ações dos agentes etc (WEYNS; OMICINI; ODELL, 2007). Assim como acontece com a dimensão institucional, ao considerar o ambiente como uma abstração primária em SMA, este passa a ser um elemento independente da presença de agentes, de suas percepções, crenças ou ações. A estrutura de um SMA com agentes, instituição e ambiente separados em abstrações primárias distintas é ilustrada na Figura 7.

A separação de agentes, instituição e ambiente em abstrações primárias distintas proporciona uma separação clara de domínios: agentes, ambiente e instituição tratam de domínios diferentes em um SMA e não precisam, necessariamente, uns dos outros para serem modelados. Para exemplificar essa separação de domínios, pode-se utilizar um cenário baseado em uma instituição de ensino, em que uma turma formada por diversos alunos e um professor se reune em uma sala de aula para estudar os conteúdos de uma disciplina. Considere-se um objeto "sala" que, no cenario usado como exemplo, é utilizado como sala de aula: pode-se descrever a sala mencionando características como localização, capacidade e infraestrutura. Por ser uma sala de aula, tal objeto tem associado a si diversos elementos de caráter institucional: o próprio fato de ser uma sala "de aula" já se explica em função da instituição (o mesmo objeto "sala" poderia ser utilizado como sala de reuniões, sala de professores etc). Além disso, existem elementos como "professores", "alunos" e "turmas", relacionados com a instituição, que têm relação com o objeto "sala". Se esses elementos relacionados com a instituição forem desconsiderados, no entanto, o objeto "sala" ainda

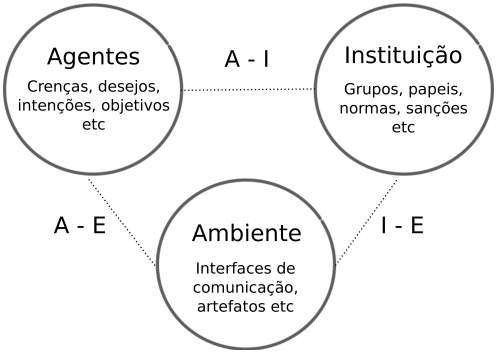


Figura 7 – Abstrações primárias em SMA (Baseado em (PIUNTI et al., 2010))

assim poderá ser descrito em função de suas características físicas (localização, capacidade, infraestrutura etc). De forma semelhante, uma "turma", reunindo diversos "alunos" e um "professor" para estudar uma "disciplina", pode ter suas aulas realizadas regularmente em uma sala. Se essa sala for diferente a cada aula, no entanto, isso não fará com que aquele grupo de pessoas deixe de ser uma turma composta por diversos alunos e um professor estudando uma disciplina. Verifica-se assim que, embora haja relação entre os elementos das dimensões institucional e ambiental, esses elementos referem-se a domínios distintos e podem ser modelados independentemente. A Figura 8 exibe um esboço dos modelos de elementos das dimensões agente, instituição e ambiente conforme o exemplo citado.

Na Figura 7, são exibidas as três dimensões e suas relações em SMA: A-I (agentes-instituição), A-E (agentes-ambiente) e I-E (instituição-ambiente) [1]. O foco, neste trabalho, concentra-se em um aspecto específico das interações I-E: o tratamento, em nível institucional, de fatos ocorridos no ambiente ou na própria instituição. Com base na teoria de Searle, é possível afirmar que os fatos institucionais ocorrem somente como consequência de fatos brutos ou de outros fatos insti-

[1] Utiliza-se aqui a letra E para referir-se ao ambiente para não haver confusão com a letra A que se refere a "agentes". A letra E foi escolhida com base na tradução para a língua inglesa da palavra "ambiente" (*environment*)

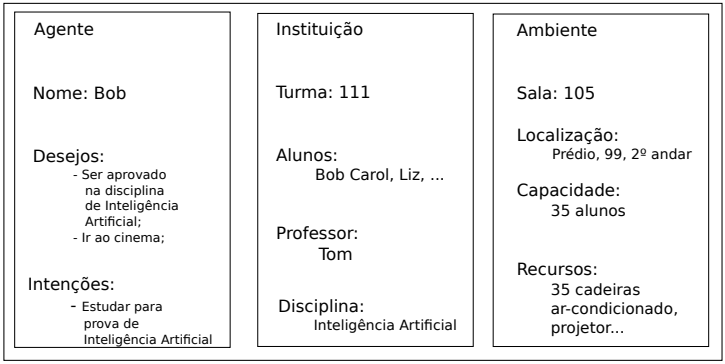


Figura 8 – Exemplo de modelo de agente, instituição e ambiente

tucionais. Por isso, em última análise, pode-se dizer que o foco do trabalho concentra-se em estabelecer um modelo para a criação de fatos institucionais em SMA a partir de fatos ocorridos no ambiente ou na instituição.

3.2 MODELO PARA A CRIAÇÃO DE FATOS INSTITUCIONAIS EM SMA

Em função da concepção de SMA baseda em três dimensões utilizada neste trabalho, ilustrada na Figura 7, as interações I-E não podem ser modeladas como parte integrante do ambiente e nem como parte integrante dos mecanismos institucionais. Se as interações I-E fossem definidas em alguma dessas dimensões, tal dimensão precisaria ter alguma representação de elementos pertencentes à outra dimensão, provocando uma intersecção entre os domínios.

Como alternativa, propõe-se que a criação de fatos institucionais (como parte das interações I-E) seja modelada de forma independente, observando ambiente e instituição "externamente", levando em consideração aspectos de cada uma das dimensões e possibilitando a criação de novas definições, não pertencentes às dimensões institucional e ambiental, porém relacionadas ao modelo em questão.

Esta seção detalha o modelo proposto para a criação de fatos institucionais a partir de fatos ocorridos no ambiente e instituição. Esse detalhamento será feito a seguir e é dividido em três partes distintas:

- Inicialmente, será feita uma série de definições básicas que

compõem o modelo conceitual proposto;

- Em seguida, será definida uma linguagem de programação que permite utilizar os elementos definidos para descrever a criação de fatos institucionais;
- Por fim, será exposta a dinâmica de um mecanismo que é capaz de trabalhar em conjunto com ambiente e instituição, executando as regras definidas segundo a linguagem.

3.2.1 Modelo conceitual

No modelo proposto, a criação de fatos institucionais a partir de fatos brutos ou de outros fatos institucionais pode ser descrita através de quatro elementos básicos:

- Eventos do ambiente e da instituição;
- Estado do ambiente e instituição;
- Expressões de conhecimento de domínio
- Regras *count-as*.

Esses elementos serão definidos a seguir.

Devido à opção de observar as estruturas ambiental e institucional "externamente", assume-se, ao propor o presente modelo, não existir garantias de que seja possível conhecê-las inteiramente. Admite-se também a possibilidade de ambiente e instituição sofrerem modificações que não podem ser previstas de antemão (como a inserção de novos componentes, modificações nos componentes existentes etc). Por esse motivo, considera-se que, entre os elementos componentes das infraestruturas ambiental e institucional, existem porções *observáveis* externamente que serão consideradas na definição do modelo. Quanto às porções não observáveis, admite-se sua existência mas não é previsto qualquer tratamento para elas. A Figura 9 ilustra essa ideia, retratando ambiente e instituição. Em cada uma dessas estruturas existe um estado e um conjunto de eventos, sendo que algumas partes do conjunto de eventos e parte do estado são observáveis. Na figura, essas porções observáveis são ilustradas em azul e as setas indicam o sentido das interações cobertas pelo modelo proposto: a origem está nas porções observáveis de eventos e estado do ambiente e da instituição e o destino está, exclusivamente, na porção observável do estado institucional.

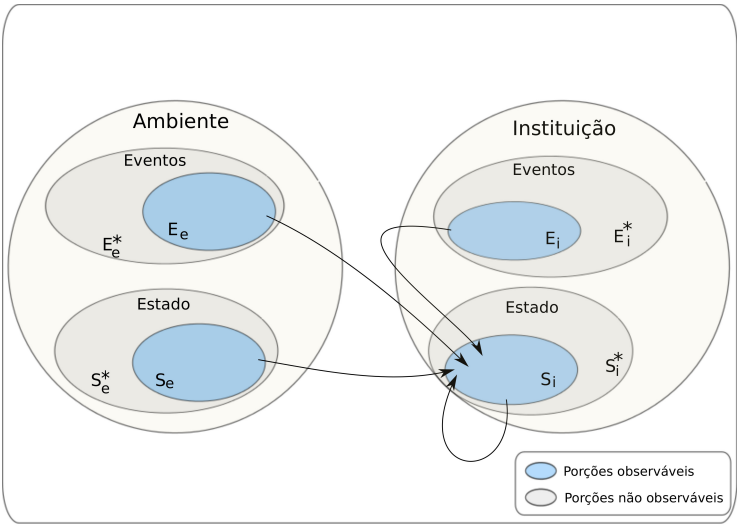


Figura 9 – Porções observáveis e não observáveis na instituição e no ambiente

Nas próximas seções, serão feitas definições mais detalhadas a respeito da concepção de eventos e estado no contexto deste trabalho.

3.2.1.1 Eventos

O conceito de evento, segundo (CASSANDRAS; LAFORTUNE, 2008), é bastante intuitivo e pode ser definido como uma ocorrência instantânea que causa uma mudança no estado do sistema, podendo ser provocada (i) por uma ação efetuada por um agente externo, como, por exemplo, o pressionamento de um botão provocando o acionamento de uma máquina (ii) de forma espontânea, por algum motivo relacionado ao ambiente, como, por exemplo, o desligamento de uma máquina devido à falta de energia ou (iii) por condições diversas que acontecem simultaneamente, como, por exemplo, a elevação da temperatura e a detecção de fumaça disparando um alarme de incêndio. Em (CURY, 2001), define-se evento como um estímulo processado pelo sistema, de forma instantânea e abrupta, refletindo ocorrências verificadas no mundo externo ou dentro do próprio sistema, causando, em geral, uma mudança interna no mesmo, nem sempre perceptível a um observador externo. Como síntese, pode-se definir evento como uma *ocorrência instantânea e abrupta, refletindo o processamento, por parte do sistema, de ocorrências internas ou externas ao mesmo. As ocorrências podem ser provocadas por um agente externo, pelo ambiente ou pela reunião de condições favoráveis. Um evento pode ter como consequência alguma mudança interna no sistema, porém tal mudança não é obrigatória e, caso venha a acontecer, não precisa ser, necessariamente, perceptível a um observador externo.*

No modelo proposto, eventos ocorridos no ambiente e na instituição podem ter consequências em nível institucional e, por isso, são definidos formalmente a seguir.

Definição 3.2.1 (Evento ambiental)**.** Sendo E_e^* o conjunto de todos os eventos possíveis de acontecerem no ambiente, define-se $E_e \subseteq E_e^*$ como o conjunto de todos os eventos observáveis no ambiente.

Definição 3.2.2 (Evento institucional)**.** Sendo E_i^* o conjunto de todos os eventos possíveis de acontecerem na instituição, define-se $E_i \subseteq E_i^*$ como o conjunto de todos os eventos observáveis na instituição.

3.2.1.2 Propriedades e estado

Neste trabalho, o *estado*, tanto do ambiente quanto da instituição, é um elemento importante. Sua definição tem relação com a ideia de *propriedade*. Os conceitos relacionados a propriedade são tratados na definição 3.2.3 com o auxílio da Figura 10. As definições 3.2.4 e 3.2.5, por sua vez, tratam da ideia de estado.

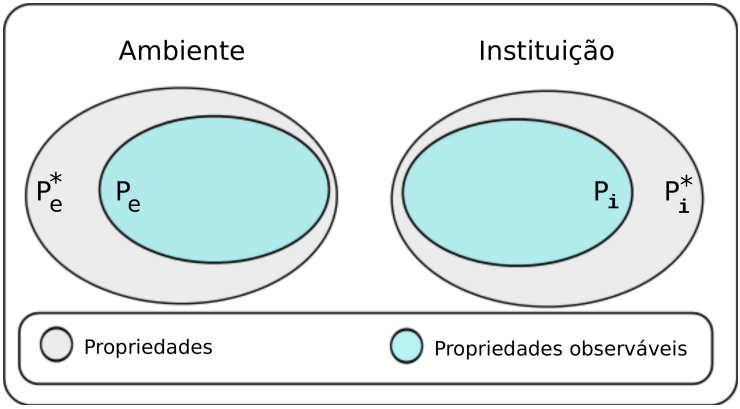


Figura 10 – Propriedades e propriedades observáveis

Definição 3.2.3 (Propriedade)**.** Define-se *propriedade*, no contexto deste trabalho, como um atributo de algum elemento pertencente à instituição (propriedade institucional) ou ao ambiente (propriedade ambiental). Assumindo a possibilidade de não se conhecer inteiramente a instituição e o ambiente, assume-se que entre todas as propriedades ambientais (P_e^*) e institucionais (P_i^*) existem *propriedades observáveis* externamente. Para tratar das propriedades observáveis ambientais e institucionais serão usados os símbolos P_e e P_i respectivamente.

Em um ambiente composto por uma única sala cujas propriedades observáveis são o estado da porta e da janela, pode-se expressar essas propriedades como $aberto(janela)$, $fechado(janela)$, $aberto(porta)$ e $fechado(porta)$.

A ideia de *propriedade* serve de base às definições relacionadas à ideia de *estado*, que será efetivamente utilizada no modelo proposto. De maneira geral, define-se que o estado do ambiente e da instituição é caracterizado por um conjunto de propriedades dos elementos que compõem essas estruturas.

Como não se pode garantir a possibilidade de observar inteiramente as propriedades do ambiente e da instituição, não se pode garantir a possibilidade de conhecer inteiramente seus estados (representados por S_e^* e S_i^* na Figura 9). Por isso considera-se a existência de *estados observáveis* no ambiente e na instituição, dados pelas propriedades observáveis dessas dimensões e representados por S_e e S_i respectivamente. Para este modelo, os estados que, de fato, têm importância, são os estados observáveis.

Definição 3.2.4 (Estado ambiental observável)**.** Define-se um estado ambiental observável s_e como um subconjunto de propriedades observáveis do ambiente (P_e). O conjunto de todos os estados possíveis de serem verificados no ambiente é representado pelo símbolo S_e e pode-se observar que $S_e = 2^{P_e}$.

Utilizando o exemplo citado anteriormente, se

$$\begin{aligned} P_e &= \{aberto(janela), fechado(janela), \\ &\quad aberto(porta), fechado(porta)\} \end{aligned}$$

então

$$\begin{aligned} S_e &= \{\{\}, \{aberto(janela)\}, \{fechado(janela)\}, \\ &\quad \{aberto(porta)\}, \{fechado(porta)\}, \\ &\quad \{aberto(janela)\}, \{fechado(janela)\}...\}. \end{aligned}$$

Observa-se que um possível estado $s_e \in S_e$ é

$$s_e = \{aberto(janela), fechado(janela)\}$$

Intuitivamente, pode-se concluir que tal estado é inconsistente pois se a propriedade $aberto(janela)$ for verdadeira, a propriedade $fechado(janela)$ será falsa. O modelo proposto, no entanto, não leva em consideração qualquer consistência em relação às propriedades. Antes disso, assume-se que o modelo sempre tratará com informações consistentes a respeito das propriedades do ambiente e da instituição.

Definição 3.2.5 (Estado institucional observável)**.** Define-se um estado institucional observável s_i como um subconjunto de propriedades observáveis do ambiente (P_i). O conjunto de todos os estados possíveis de serem verificados na instituição é representado pelo símbolo S_i e pode-se observar que $S_i = 2^{P_i}$.

Definição 3.2.6 (Expressão de conhecimento de domínio (*domain kno-*

wledge statement)). Uma expressão de conhecimento de domínio expressa informações que são necessárias para modelar a criação de fatos institucionais mas que não são relacionadas exclusivamente ao ambiente nem à instituição. Para exemplificar, pode-se utilizar novamente o cenário de uma universidade em que o evento de um agente entrar em uma sala de aula faz com que este assuma o papel de aluno. Essa universidade pode ter várias salas, alguma delas sendo utilizadas como salas de aula, outras como salas de professores e outras como salas de reuniões. Nesse cenário, o ambiente tem, entre seus elementos, as diversas salas e foge de seu domínio a forma como estas serão utilizadas. Ou seja, a ideia de *sala de aula* não pertence ao domínio do ambiente. Por outro lado, a ideia de *sala de aula* é inerente à instituição e não depende necessariamente do elemento físico *sala* para ser expressa. Suponha-se que nesse cenário existam duas salas: a sala número 1 é uma sala de aula e a sala número 2 é uma sala de reuniões. Nesse caso, um agente entrando na sala número 1 assume o papel de aluno enquanto um agente entrando na sala número 2 não tem implicação institucional. Verifica-se assim que, para poder aplicar a regra, é necessário poder modelar a ideia de que a sala número 1 é uma sala de aula. Essa ideia é uma expressão de conhecimento de domínio (DKS). Um conjunto de sentenças DKS é chamado *Domain Knowledge Base* (DKB) .

3.2.2 Linguagem de programação

A partir das definições feitas anteriormente, é possível apresentar uma linguagem, baseada no modelo descrito neste capítulo, para programar *constitutive rules*, cuja sintaxe é especificada na Figura 11. A gramática ilustrada na Figura 11 representa um programa *count-as* (*count-as-program*). Um programa *count-as* é composto por (i) um conjunto de *constitutive rules* (representadas, na gramática pelo elemento `countas_stat`) e (ii) um conjunto opcional de expressões DKS, representadas, na gramática pelo elemento `domain_knowledge_stat`.

Cada `countas_stat`, à semelhança das *constitutive rules* definidas na teoria de Searle, tem a forma *X count as Y in C* e, por isso, é composta por um fato bruto (representado, na gramática, pelo elemento `termX`), relacionado a um fato institucional (representado, na gramática, pelo elemento `termY`) por um operador `count-as`. Além disso, cada `countas-stat` pode ter um contexto em que é aplicável, definido através do elemento `context`, relacionado à regra pelo ope-

```
count_as_program      ::= (domain_kldge_base)?countas_stat+
domain_kldge_base     ::= 'domain_knowledge_base:'(domain_knowledge_stat'.')+
domain_knowledge_stat ::= predicate
countas_stat          ::= termX 'count-as' termY('in' context)?'.'
termX                 ::= event|state
event                 ::= '+'structure
state                 ::= '*'formula
termY                 ::= predicate(','predicate)*
context               ::= formula
predicate             ::= ATOM(pred_term(','pred_term)*)?annotation?
pred_term             ::= VAR|TERM_NULL|STRING|predicate
annotation            ::=list
list                  ::='['list_term(','list_term)*']'
list_term             ::= list|arithm_expr|STRING
formula               ::= log_expr_trm('|'formula)?
log_expr_trm          ::= log_expr_factor('&'log_expr_trm)?
log_expr_factor       ::= NEGATION log_expr_factor|rel_expr
rel_expr              ::= (arithm_expr|STRING)
                          ('<'|'<='|'>'|'>='|'=='|'\\=='|'=')
                          (arithm_expr|STRING))?
arithm_expr           ::= arithm_expr_trm(('+'|'-')arithm_expr)?
arithm_expr_trm       ::= aritm_expr_factor((arithm_expr_op)arithm_expr_trm)?
arithm_expr_op        ::='*'|'/'|INTDIV|INTMOD
arithm_expr_factor    ::= arithm_expr_simple(('**')arithm_expr_factor)?
arithm_expr_simple    ::= VAR|DIGIT+|pred|'('log_expr')'
STRING               ::= '"'ATOM|VAR'"'
NEGATION             ::= 'not'
INDTIV               ::= 'div'
INTMOD               ::= 'mod'
ATOM                 ::= ('a'..'z')('a'..'z'|'A'..'B'|DIGIT|'_'|'.')*
VAR                  ::= ('A'..'Z')('a'..'z'|'A'..'B'|DIGIT|'_'|'.')
DIGIT                ::= '0'..'9'
TERM_NULL            ::= '_'
```

Figura 11 – Gramática para a linguagem proposta

rador `in`. O contexto é um elemento opcional na sintaxe de uma regra pois podem existir regras que devem ser aplicadas em qualquer contexto. Uma definição formal para um programa *count-as* é dada a seguir.

Definição 3.2.7 (Programa *count-as*)**.** Um programa *count-as* é composto pela tupla $\langle R_e, R_s, D \rangle$, onde:

- R_e é um conjunto de regras *count-as* que tratam eventos ocorridos no ambiente e na instituição;
- R_s é um conjunto de regras *count-as* que tratam de propriedades verificadas no ambiente e na instituição;
- D é um conjunto de expressões de conhecimento de domínio (DKS) que definem aspectos relacionados ao domínio tratado pelo programa *count-as*.

No modelo proposto neste trabalho, os fatos brutos, representados pelo elemento `termX` na gramática, podem ser eventos ou estados verificados no ambiente ou na instituição. Os fatos institucionais, por sua vez, são estados da instituição. Em função dos diferentes tipos de termo X, são definidos dois tipos de regras *count-as*: regras *event-count-as*, em que o `termX` é composto por um elemento `event` e regras *state-count-as*, em que o `termX` é composto por um elemento `state`. Estes dois tipos de regras *count-as* são definidos a seguir.

3.2.3 Regras *count-as*

Regras *event-count-as* e *state-count-as* são regras que definem a criação de fatos institucionais a partir de eventos e estado, respectivamente. Essas regras serão definidas formalmente nesta seção. Antes de fazer essas definições, no entanto, serão analisados alguns aspectos relacionados à utilização de eventos e estados como fatos brutos.

3.2.3.1 Discussão sobre processamento de eventos e estados

Como é possível observar na Tabela 1, alguns modelos consideram que fatos institucionais podem ser gerados por eventos e alguns modelos consideram que esses fatos podem ser gerados em função de estados que passam a ser verdadeiros no sistema. Não se verificou entre os modelos analisados na seção 2.2, no entanto, qualquer modelo que

considere tanto eventos quanto estados como geradores de fatos institucionais. Apesar de não haver justificativa clara nos modelos estudados, existe uma aparente ideia de que a utilização de eventos e estados no mesmo modelo é redundante. Através dos estudos realizados, no entanto, verificou-se que ambas as abordagens podem ser abrigadas no mesmo modelo. Justifica-se isso por três motivos principais: (i) conhecimento parcial da estrutura ambiental e institucional, (ii) poder de expressão de regras *count-as* e (iii) questões relacionadas à concorrência e ordenação de eventos.

Considerando a ideia de que o modelo proposto neste trabalho busca manter generalidade quanto a aspectos específicos de implementação das infraestruturas ambiental e institucional (que são as estruturas com quem o modelo irá interagir), admite-se a possibilidade de que o conhecimento a respeito do modelo do ambiente e instituição, bem como de suas implementações, seja apenas parcial. Isso implica na possibilidade de não conhecer-se com clareza a relação entre eventos e as consequentes alterações de estado verificadas no sistema. Retomando um exemplo citado anteriormente, considere-se o cenário de uma universidade em que o evento gerado pelo fato de um agente entrar em uma sala de aula tenha como consequência, em nível institucional, a adoção do papel de aluno por parte do agente. Uma outra consequência relativamente óbvia do mesmo evento é uma alteração no estado do sistema: antes da ocorrência do evento, o agente estava fora da sala de aula e após a ocorrência, o agente passou a estar dentro da sala. Poderia-se, a partir disso, definir uma regra *event-count-as* definindo que o evento de o agente entrar na sala *count-as* o agente passar a ter o papel de aluno. Suponha-se, no entanto, que o ambiente não produza qualquer evento no momento em que o agente entra na sala. Isso implica na impossibilidade de detectar o momento exato em que o agente entra na sala. Adicionalmente, suponha-se que o sistema permita verificar o estado, detectando que o agente, em determinado momento, passa a estar dentro da sala. Nesse caso, mesmo sendo impossível definir uma consequência institucional para o evento de o aluno entrar na sala, é possível definir uma consequência para o estado de o aluno se encontrar dentro da sala.

Além disso, podem existir situações em que diversos eventos produzem um mesmo estado ou em que o projetista não tem conhecimento total dos eventos que produzem determinados estados. Regras *state-count-as* podem ser mais expressivas pois podem ser acionadas por estados que são produzidos por diversos eventos, sem necessitar conhecer ou exprimir tais eventos.

Outro aspecto que torna conveniente, em determinadas situações, a utilização de regras acionadas por estados, tem relação com a ordenação de eventos e o processamento concorrente. Para exemplificar essa questão, suponha-se um cenário em que um evento e_1 produz um estado α e um evento e_2 produz um estado β. Suponha-se ainda que, nesse cenário, a ocorrência do evento e_2 tenha como consequência institucional o estado γ se o estado α for verdadeiro. Essa regra pode ser expressa como e_2 *count-as* γ *in* α. Nesse caso, para que a regra seja executada, o evento e_1, que produz o estado α, precisa acontecer antes do evento e_2. Suponha-se, no entanto, que o evento e_1 ocorra e, quando e_2 ocorrer, devido a motivos quaisquer, o estado α, ainda não tornou-se verdadeiro no sistema. Nesse caso, a regra não será processada, pois por ocasião da ocorrência do evento e_2, o estado α não é verdadeiro apesar de o evento e_1 já ter ocorrido. Apesar de essas questões serem questões relacionadas a detalhes de implementação, em virtude da ordenação de eventos e do processamento de estados consequentes de eventos, considera-se que o modelo pode abrigar tais questões pois certos cenários reais apresentam tal particularidade. Voltando ao cenário hipotético de uma sala de aula, suponha-se que em tal cenário exista uma regra que diga que o fato de um professor entrar na sala aula tem o significado institucional de que a aula iniciou caso exista algum aluno na sala. Nesse cenário, suponha-se que que o professor entre na sala de aula e, no instante imediatamente posterior, o primeiro aluno entre na sala. Nesse caso, o processamento do evento de o professor entrar na sala não produziria o efeito institucional desejado. Uma alternativa para isso seria definir uma regra *state-count-as* definindo que o estado de o professor estar na sala e de haver um aluno na mesma sala tenha o significado institucional de aula iniciada, independente da ordem em que professor e aluno entraram na sala.

Apesar disso, entretanto, existem situações em que regras *event-count-as* podem ser mais apropriadas. Assim como em algumas situações pode não ser possível determinar o evento que produz um determinado estado no sistema, existe a possibilidade de, em outras situações, não ser possível determinar o estado provocado por um evento. Esse é um exemplo de situação em que regras *event-count-as* são convenientes. Além disso, o processamento de regras *event-count-as* mostra-se mais eficiente, pois eventos são processados pontualmente por ocasião de sua ocorrência, enquanto o estado precisa ser constantemente monitorado.

3.2.4 Regras *event-count-as* e *state-count-as*

Considerando as questões mencionadas anteriormente, a seguir são definidas regras para a criação de fatos institucionais a partir de eventos e estados.

Definição 3.2.8 (Regra *event-count-as*)**.** Uma regra *event-count-as* é definida pela tupla $\langle b_f, i_f, c \rangle$ onde:

- $b_f \in E_e \cup E_i$ é um evento do ambiente ou da instituição correspondente a um fato bruto;
- i_f é o fato institucional, composto por um conjunto de propriedades que a instituição deve passar a possuir em função da aplicação da regra;
- c é uma fórmula lógica composta por predicados pertencentes aos conjuntos P_e e P_i e indica os estados observáveis ambiental e institucional que devem ser verdadeiros para que a regra seja aplicada.

As regras *event-count-as* são regras que definem um novo estado institucional em função de eventos ocorridos no ambiente ou na instituição. Relacionando a definição à gramática ilustrada na Figura 11, uma regra *event-count-as* é uma regra *count-as* `countas_stat` cujo fato bruto b_f (`termX`) é um evento (`event`). Conforme definido, o fato institucional i_f é estado institucional que, em função da ocorrência do evento b_f, deve passar a ser verdadeiro na instituição. Como exemplo, pode-se definir, em um sistema, que, ao entrar em uma determinada sala de aula, um agente assuma o papel de aluno caso esteja matriculado em um curso específico e caso o evento ocorra às 10 horas da manhã de sexta-feira. Nesse caso, o evento específico de entrar na sala é o fato bruto b_f que tem como consequência institucional i_f o fato de o agente passar a exercer o papel de aluno. O contexto, definido no termo c da regra, é formado por uma combinação de estados ambiental e institucional que devem ser verdadeiros para que a regra seja aplicada. Utilizando o mesmo exemplo citado anteriormente, o contexto c é composto (i) por um estado ambiental, que condiciona a aplicação da regra ao fato de o evento b_f acontecer às 10h da manhã de sexta-feira e (ii) por um estado institucional, que é o fato de o agente estar matriculado em um curso. Um exemplo para a sintaxe dessa regra é exibido na Figura 12.

```
+entrouNaSala(101)[agente(Ag)]
count-as
  desempenhandoPapel(Ag, aluno)
in
  horaAtual("10 am") &
  diaAtual(sextaFeira)&
  matriculado(Ag, inteligênciaArtificial).
```

Figura 12 – Exemplo de regra *event-count-as*

Conforme a gramática ilustrada na Figura 11 e conforme o exemplo ilustrado na Figura 12, uma regra *event-count-as* é iniciada pelo caractere "+". Depois disso, o elemento `entrouNaSala(101)`, correspondente ao fato bruto, define o evento que, ao ocorrer, provoca o acionamento da regra. Para isso, conforme a gramática, é utilizado um elemento *structure*. No caso do exemplo da Figura 12, o evento é descrito por um nome (`entrouNaSala`) e tem um argumento (`101`), indicando a sala em que o agente entrou. Além disso, a descrição do evento possui uma *anotação* indicando o agente que provocou sua ocorrência. Como é possível observar no exemplo, esse agente é representado por pelo termo `Ag`, que, por ser iniciado por letra maiúscula, consiste em uma variável. No exemplo, o fato institucional i_f, representado pelo predicado `desempenhandoPapel(Ag, aluno)`, indica que, após a aplicação da regra, a instituição passará a ter a propriedade `desempenhandoPapel`, sendo que o argumento Ag será substituído por um termo representando o agente que provocou a ocorrência do evento. A regra será aplicada caso o evento ocorra às 10 horas da manhã de sexta-feira (estado do ambiente) e se o agente que provocou a ocorrência do evento estiver matriculado na disciplina de Inteligência Artificial (estado da instituição) [2] .

Definição 3.2.9 (Regra *state-count-as*)**.** Uma regra count-as $r_s \in R_s$ é definida pela tupla $\langle b_f, i_f, c \rangle$ onde:

- b_f é um fato bruto que consiste em uma fórmula lógica composta por predicados pertencentes aos conjuntos P_e e P_i e indica os estados observáveis ambiental e institucional que devem ser verdadeiros para que a regra seja ativada;

[2] Alguns conceitos, como estrutura, termo, predicado, entre outros, são relacionados à linguagens de programação baseados em lógica e, em específico, à linguagem de programação de agentes Jason. Mais detalhes a respeito desses conceitos estão descritos no Apêndice A.

- i_f é o fato institucional, composto por um conjunto de propriedades que a instituição deve passar a possuir em função da aplicação da regra;
- c é uma fórmula lógica composta por predicados pertencentes aos conjuntos P_e e P_i e indica os estados observáveis ambiental e institucional que devem ser verdadeiros para que a regra seja aplicada.

As regras *state-count-as* são regras que definem um novo estado institucional em função do estado da instituição e do ambiente. Relacionando a definição à gramática ilustrada na Figura 11, uma regra *state-count-as* é uma regra *count-as* `countas_stat` cujo fato bruto b_f (`termX`) representa uma combinação de estados do ambiente ou da instituição (`state`). À semelhança das regras *event-count-as*, nas regras *state-count-as*, o fato institucional i_f é estado institucional que deve passar a ser verdadeiras na instituição caso os estados definidos em b_f sejam verdadeiros no ambiente e na instituição. Também de forma semelhante às regras *event-count-as*, o contexto, definido no termo c das regras *state-count-as*, é um conjunto formado por uma combinação de estados ambiental e institucional. A Figura 13 ilustra um exemplo de regra *state-count-as*.

```
*alunosNaSala(101,X) & X >= 30 &
 agenteNaSala(Agente,101) &
 desempenhandoPapel(Agente,Papel) &
 Papel = professor &
 horaAtual("10 am")
count-as
  aulaEmAndamento(inteligênciaArtificial)
in
  not (aulaEmAndamento(inteligênciaArtificial)).
```

Figura 13 – Exemplo de regra - estado

Conforme ilustrado na Figura 13 e conforme a gramática ilustrada na Figura 11, uma regra *state-count-as* é iniciada pelo caractere "$*$" seguido pelo "termo X" da regra, composto por uma fórmula lógica que expressa o estado de que deve ser verdadeiro para que a regra seja executada. No caso da regra ilustrada pela Figura 13, esse estado é dado por quatro propriedades:

- `alunosNaSala(X,Y)` - indica que existem Y alunos na sala X;

- `agenteNaSala(X,Y)` - indica que o agente X está na sala Y;
- `desempenhandoPapel(X,Y)` - indica que o agente X está desempenhando o papel Y;
- `horaAtual(X)` - indica que a hora atual é X;

A regra ilustrada na Figura 13 será executada caso existam trinta ou mais alunos dentro da sala 101 e se houver um agente desempenhando papel de professor dentro da sala 101 às 10 horas da manhã.

O elemento `termY` da regra descreve propriedades que devem passar a ser verdadeiras na instituição em virtude da aplicação da regra. No exemplo ilustrado pela Figura 13, a execução da regra deve fazer com que a instituição passe a ter a propriedade `aulaEmAndamento(inteligênciaArtificial)`, indicando que a aula de Inteligência Artificial está em andamento. A forma como a estrutura institucional procederá para alcançar tal estado tem relação direta com a implementação de tal estrutura, sendo assim um detalhe de arquitetura que será discutido posteriormente.

O contexto c, correspondente, na gramática, ao elemento `termC`, é expresso por uma fórmula lógica que determina que a regra só será executada caso a propriedade `aulaEmAndamento(inteligênciaArtificial)` não seja verdadeira na instituição.

3.2.5 *Count-as Engine*

Tendo definido os elementos componentes do modelo e a linguagem, é possível definir de que forma esses elementos podem ser utilizados para dar significado institucional a fatos brutos ou a outros fatos institucionais. No presente modelo, é utilizado o nome *Count-as Engine* (CE) para se referir à estrutura projetada para interpretar regras *count-as*. O CE opera lado a lado com as estruturas ambiental e instutucional em um SMA, recebendo informações a respeito da ocorrência de eventos e alterações no estado observável dessas estruturas, processando essas informações e verificando as consequências institucionais desse processamento. Essas consequências institucionais são propriedades de devem passar a ser verdadeiras na instituição. O CE verifica quais são essas propriedades e disponibiliza essa informação à instituição, que deve processá-las de maneira adequada.

Para descrever o funcionamento do CE, será usada semântica operacional. Por isso, inicialmente será definida a configuração do CE

(na Definição 3.2.10) para, em seguida, definir suas possíveis transições de estado.

Definição 3.2.10 (Configuração do *Count-as Engine*)**.** A configuração do CE é dada pela tupla $\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle$ onde:

- R_e é um conjunto de regras *event-count-as*;
- R_s é um conjunto de regras *state-count-as*;
- D é um conjunto de expressões DKS;
- $\mathcal{E}$ é uma fila de eventos $e \in E_e \cup E_i$
- $\mathcal{N} \in S_e$ é uma representação do estado observável atual do ambiente, dado por todas as suas propriedades observáveis;
- $\mathcal{I} \in S_i$ é uma representação do estado observável atual da instituição, dado por todas as suas propriedades observáveis;
- $\mathcal{T}$ é um conjunto de propriedades, resultantes da aplicação das regras *count-as*, que devem tornar-se verdadeiras na instituição

A configuração inicial do CE é dada pela tupla $\langle R_e, R_s, D, \emptyset, \emptyset, \emptyset, \emptyset \rangle$ pois, inicialmente, não existem eventos a serem processados, não existem propriedades conhecidas no ambiente e na instituição e, consequentemente, não há qualquer resultado de processamento. A partir do início da execução, o CE vai sendo alimentado com os eventos ocorridos no ambiente e na instituição, e com informações sobre as propriedades observáveis do ambiente e da instituição. Como as informações sobre eventos ($\mathcal{E}$) e sobre propriedades ($\mathcal{N}$ e $\mathcal{I}$) são provenientes do meio externo ao CE, assume-se que tais informações correspondam à realidade verificada no meio externo. A partir dessa representação, o CE realiza o processamento necessário, verificando propriedades que a instituição deve passar a ter e colocando essas propriedades em $\mathcal{T}$. As propriedades pertencentes a $\mathcal{T}$ são disponibilizadas para que a dimensão institucional as processe. Espera-se assim que cada propriedade em $\mathcal{T}$ passe a pertencer a P_i. Além disso, o CE e possibilita a inclusão ou remoção de regras *count-as* e expressões DKS. Essas operações são descritas a seguir.

Definição 3.2.11 (Adição e remoção regra *count-as*)**.** Durante a execução do programa, novas regras podem ser adicionadas ou removidas.

No caso de adição de uma regra *event-count-as* r_e, o processamento realizado é o seguinte:

$$\frac{r_e \quad r_e \notin R_e}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle \longrightarrow \langle R_e \cup r_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle}$$

De forma semelhante, no caso de adição de uma regra *state-count-as* r_s, o processamento realizado é o seguinte:

$$\frac{r_s \quad r_s \notin R_s}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle \longrightarrow \langle R_e, R_s \cup r_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle}$$

No caso de remoção de uma regra *event-count-as* r_e, o processamento realizado é o seguinte:

$$\frac{r_e \quad r_e \in R_e}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle \longrightarrow \langle R_e \setminus r_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle}$$

De forma semelhante, no caso de remoção de uma regra *state-count-as* r_s, o processamento realizado é o seguinte:

$$\frac{r_s \quad r_s \in R_s}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle \longrightarrow \langle R_e, R_s \setminus r_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle}$$

Definição 3.2.12 (Adição ou remoção de expressão DKS)**.** Durante a execução do programa, novas expressões DKS podem ser adicionadas ou removidas.

No caso de adição de uma expressão DKS d, o processamento realizado é o seguinte:

$$\frac{d \quad d \notin D}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle \longrightarrow \langle R_e, R_s, D \cup d, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle}$$

No caso de remoção de uma expressão DKS d, o processamento realizado é o seguinte:

$$\frac{d \quad d \in D}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle \longrightarrow \langle R_e, R_s, D \setminus d, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \rangle}$$

Definição 3.2.13 (Processamento de evento)**.** Considerando que a fila $\mathcal{E}$ de eventos a serem processados não esteja vazia, seja $\mathsf{head}(\mathcal{E})$ uma função que retorne o primeiro evento e da fila $\mathcal{E}$ e seja uma função $\mathsf{tail}(\mathcal{E})$ uma função que retorne a fila sem o seu primeiro elemento. Adicionalmente, seja θ uma substituição em que todas as variáveis de

b_f são substituidas por termos correspondentes em $\mathsf{head}(\mathcal{E})$.

Caso exista uma regra *event-count-as* cujo fato bruto b_f, através da aplicação de uma substituição θ, seja igual ao evento retornado por $\mathsf{head}(\mathcal{E})$, essa regra será aplicada se o estado de interesse definido no termo c for consequência lógica do estado do ambiente e instituição .

Caso essas condições sejam atendidas, a regra será executada e o fato institucional i_f será adicionado à fila de resultados $\mathcal{T}$. Formalmente:

$$\frac{\langle b_f, i_f, c\rangle \in R_e \qquad b_f\theta = \mathsf{head}(\mathcal{E}) \qquad \mathcal{N} \cup \mathcal{I} \cup D \models c}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle \longrightarrow \langle R_e, R_s, D, \mathsf{tail}(\mathcal{E}), \mathcal{N}, \mathcal{I}, \mathcal{T} \cup i_f\rangle}$$

Definição 3.2.14 (Processamento de estado)**.** Cada regra $r_s = \langle b_f, i_f, c\rangle \in R_s$ em que o fato bruto b_f e o contexto c correspondam a um estado verdadeiro no ambiente e instituição é executada da seguinte forma:

$$\frac{\langle b_f, i_f, c\rangle \in R_s \qquad \mathcal{N} \cup \mathcal{I} \cup D \models b_f \qquad \mathcal{N} \cup \mathcal{I} \cup D \models c}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle \longrightarrow \langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \cup i_f\rangle}$$

Definição 3.2.15 (Entrega de operações à aplicação)**.** Seja consme_{pt} uma função pela qual a plataforma institucional *pt* consuma um elemento do CE. Se $\mathcal{T} \neq \emptyset$, a qualquer instante a instituição pode retirar de $\mathcal{T}$ o primeiro elemento.

$$\frac{\mathcal{T} \neq \emptyset \qquad t = \mathsf{head}(\mathcal{T}) \qquad \mathsf{consme}_{pt}(t)}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle \longrightarrow \langle \mathcal{E}, \mathcal{SE}, \mathcal{SI}, R_e, R_s, \mathcal{T} \setminus t\rangle}$$

3.3 CONCLUSÕES

O modelo proposto neste capítulo apresenta uma concepção sobre como fatos verificados no ambiente e na instituição podem ter um significado em nível institucional. Essa concepção tem como uma de suas bases a teoria da construção da realidade social de John Searle. Uma segunda base para o modelo proposto é a concepção de SMA como um sistema aberto composto, basicamente, por três abstrações primárias: agentes, ambiente e instituição. Sob essa perspectiva, optou-se por modelar as interações I-E fora das estruturas ambiental e institucional, mantendo assim a independência conceitual e funcional entre

elas.

A partir disso, o modelo define que os fatos institucionais podem ter origem em eventos ou em estados específicos verificados no ambiente e na instituição, sendo que cada uma das abordagens apresenta vantagens e desvantagens em situações específicas. Diversos detalhes a respeito de eventos e estados foram definidos (eventos e estados observáveis, eventos e estados de interesse etc). Esses elementos mostraram-se suficientes para a criação de regras *count-as*, à semelhança das *constitutive rules* propostas por Searle, definindo significado institucional para fatos verificados no ambiente e na instituição. A partir disso, foi definida uma linguagem de programação para descrever tais regras e também foi definido um mecanismo chamado *Count-as Engine* para interpretar tais regras.

O modelo, composto pelos elementos citados, foi definido utilizando formalismos que abstraem aspectos relacionados a arquitetura e implementação. Para ser aplicado, no entanto, o modelo necessita de uma arquitetura que o suporte, bem como de uma implementação para essa arquitetura. No próximo capítulo, será apresentada uma arquitetura para fornecer suporte a esse modelo, definindo mecanismos que venham a viabilizar o processamento das regras *count-as*, conforme o modelo apresentado.

4 ARQUITETURA

O capítulo anterior descreveu o *modelo* proposto para tratar da criação de fatos institucionais a partir de fatos (eventos e estado) ocorridos no ambiente e na instituição em SMA abertos. Foram definidos, para isso, os elementos essenciais para a composição do modelo, uma linguagem de programação e a dinâmica envolvendo tais elementos em um programa de computador que faça essa interação.

Este capítulo, por sua vez, apresenta uma proposta de *arquitetura* que viabiliza o funcionamento do modelo proposto. Esta é *uma* arquitetura possível, sendo que outras arquiteturas podem ser propostas sobre o modelo exposto no Capítulo 3. De forma resumida, o funcionamento da arquitetura é o seguinte: o *Count-as Engine* recebe informações a respeito de eventos e propriedades verificados nas estruturas ambiental e institucional, bem como regras *count-as* descritas em um programa *count-as*. Com essas informações, o *Count-as Engine* faz o processamento das regras e disponibiliza os resultados à infraestrutura institucional. Essa visão resumida é ilustrada na Figura 14.

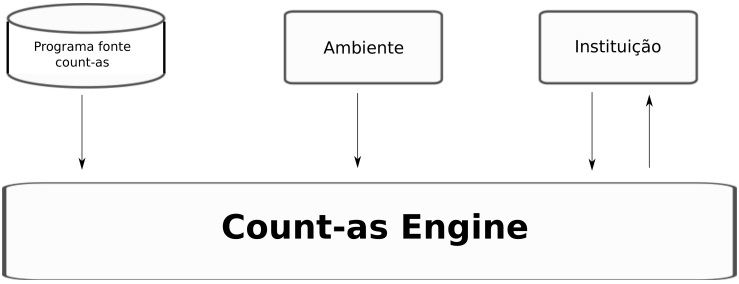


Figura 14 – Arquitetura - perspectiva resumida

A Figura 15 fornece uma visão ampla da arquitetura, dividida, essencialmente, em três camadas:

- Camada *count-as*. Incorpora os mecanismos necessários para a interpretação das regras *count-as* utilizando algumas informações externas: (i) um código fonte de um programa *count-as* e (ii) informações de eventos e propriedades das camadas ambiental e institucional, as quais são obtidas por meio de interfaces. Essa camada foi projetada para manter independência com relação aos mecanismos presentes nas outras camadas. Ou seja, nessa

camada, não há preocupação com relação à forma como as informações serão trocadas com o meio externo. Ao contrário, essa preocupação é delegada à camada de interfaces.

- Camada de interfaces. Incorpora mecanismos necessários para fazer interface entre a camada *count-as* e as infraestruturas ambiental e institucional. Assumindo a independência entre a camada *count-as* e as estruturas ambiental e institucional, assume-se também que os componentes dessas camadas não precisam ser projetados para funcionar em conjunto. As interfaces são mecanismos que conectam essas duas camadas, preservando a independência entre a aplicação e a camada *count-as* e, ao mesmo tempo, conectando-as, (i) fazendo com que eventos e propriedades sejam informados à camada count-as de maneira apropriada e (ii) obtendo da camada *count-as* os resultados do processamento das regras *count-as* e aplicando-os à infraeastrutura institucional.

- Camada de aplicação. Composta pelos elementos que compõem o SMA. Em especial, no contexto da arquitetura aqui proposta, a camada de aplicação abriga as estruturas ambiental e institucional.

O funcionamento e a ligação entre essas três camadas será descrita nas seções que seguem.

4.1 CAMADA *COUNT-AS*

A camada *count-as* realiza a interpretação das regras *count-as*, implementando os elementos do modelo proposto no Capítulo 3. Essa camada incorpora um analisador sintático (*parser*) para a linguagem proposta, bem como os mecanismos necessários para processar eventos e estados. Nessa camada, entretanto, não há qualquer preocupação quanto à captação dos eventos ocorridos e das mudanças de estado nas estruturas ambiental e institucional nem quanto à aplicação do resultado da interpretação das regras à instituição. Essas questões são tratadas pela camada de interface. A camada *count-as*, exibida na Figura 15 juntamente com o restante da arquitetura proposta, é colocada em destaque na Figura 16.

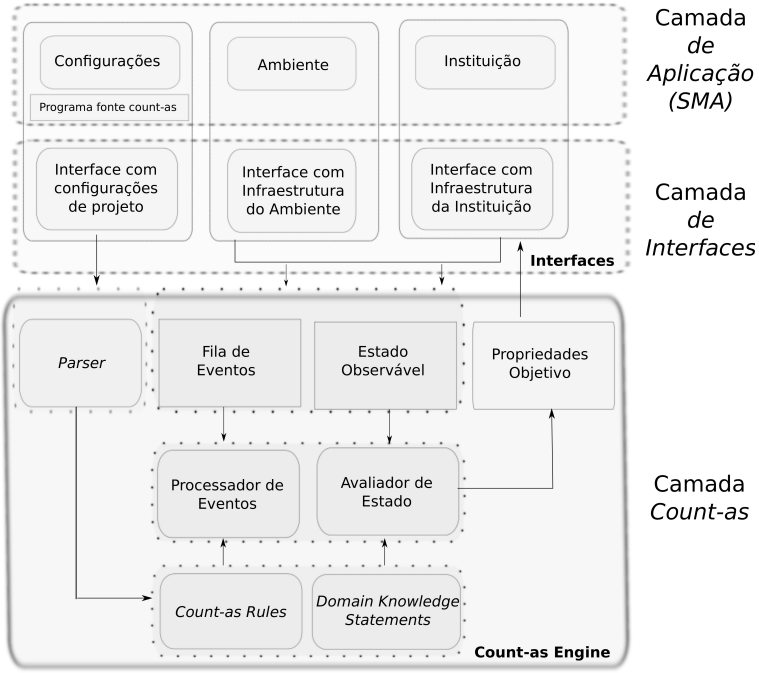


Figura 15 – Arquitetura - *Count-as Engine*

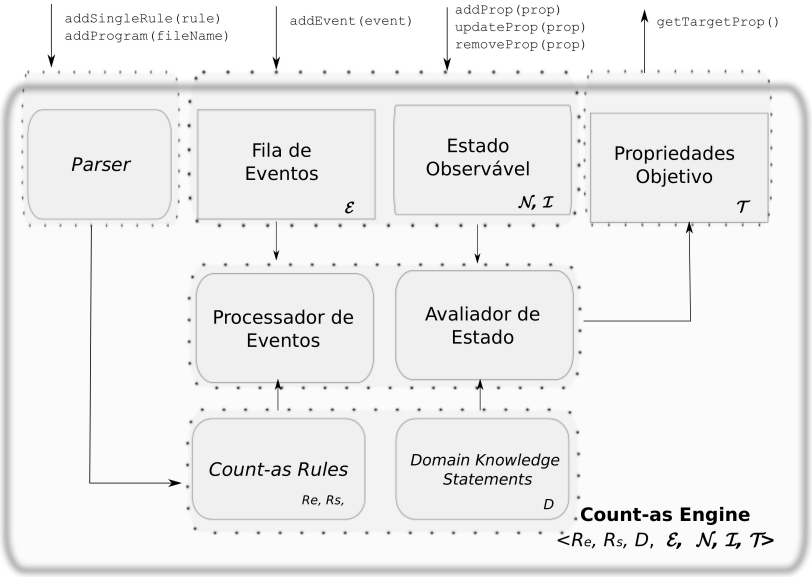


Figura 16 – Arquitetura - visão detalhada do *Count-as Engine*

4.1.1 Visão geral

A camada *count-as* é composta pelo *Count-as Engine* (CE), que é o conjunto de estruturas que, trabalhando em conjunto, realizam a interpretação das regras *count-as*. O CE recebe informações do restante da aplicação de três maneiras diferentes: (i) através da carga de um programa *count-as*, (ii) através da informação, por parte das interfaces com ambiente e instituição, de eventos ocorridos e (iii) através da informação, por parte das interfaces com ambiente e instituição, de alterações em propriedades observáveis nessas estruturas. Como saída, o CE disponibiliza as *propriedades objetivo* que são propriedades que devem passar a ser verdadeiras na instituição como resultado do processamento das regras.

De forma resumida, o CE possui uma representação de um programa *Count-as* (definição 3.2.7). Os eventos ocorridos no ambiente e instituição, recebidos da camada de interface, são armazenados em uma fila que é continuamente inspecionada pelo processador de eventos. Se houver eventos a serem processados (armazenados na fila de eventos), estes serão retirados da fila e processados pelo *Processador de*

Eventos. Além disso, o CE possui uma representação das propriedades observáveis do ambiente e da instituição. O *Avaliador de Estado* continuamente avalia essas propriedades. O resultado dos processamentos feitos pelo Processador de Eventos e pelo Avaliador de Estados é armazenado no conjunto de *Propriedades Objetivo*, ficando assim, à disposição da aplicação.

Os processamentos realizados pelo CE podem ser divididos em dois grupos: processamentos internos, que não dependem diretamente da interação com elementos externos ao CE, e processamentos externos, que envolvem algum tipo de interação com elementos externos ao CE. Os processamentos internos são os processamentos de regras *count-as* e são descritos nas seções 4.1.2 e 4.1.3. Os processamentos externos, por sua vez, são aqueles que se realizam interagindo com a camada de interfaces e são descritos nas seções 4.1.4, 4.1.5, 4.1.6 e 4.1.7.

4.1.2 Processamento de eventos

Conforme mencionado anteriormente, os eventos ocorridos no ambiente e na instituição são recebidos da camada de interface e armazenados em uma fila ($\mathcal{E}$). O CE controla constantemente essa fila e, quando esta possui um ou mais elementos, o primeiro deles é retirado e processado. Esse processamento se dá da seguinte maneira: o CE verifica se existe, no conjunto de regras R_e, uma regra $r_e = \langle b_f, i_f, c\rangle$ cujo fato bruto b_f corresponda ao evento retirado da fila. Se houver, o CE verifica o contexto c definido na regra. Se o contexto estabelecido na regra estiver de acordo com o estado observável do ambiente e instituição, composto pelas propriedades armazenadas nos conjuntos $\mathcal{N}$ e $\mathcal{I}$, isso indica que a regra é aplicável. Sendo a regra aplicável, o CE adiciona o fato institucional i_f da regra à fila de Propriedades Objetivo ($\mathcal{T}$). Esse processamento corresponde à Definição 3.2.13. O Algoritmo 1, expressa, de forma resumida, o processamento de eventos pelo CE.

Algoritmo 1: Algoritmo para processamento de evento

```
e = head(E)
se existe regra r_e = ⟨b_f, i_f, c⟩ ∈ R_e onde e = b_f então
    se N ∪ I ⊨ c então
        adiciona i_f a T
    fim
fim
```

4.1.3 Processamento de estados

O processamento de estados feito pelo CE tem por objetivo identificar estados do ambiente e instituição que tenham efeito em nível institucional. Conforme mencionado anteriormente, o CE possui uma representação das propriedades observáveis do ambiente ($\mathcal{N}$) e da instituição ($\mathcal{I}$), a partir das quais é possível identificar seu estado. Considerando que esses estados podem se modificar constantemente, sua avaliação precisa ser contínua. Por esse motivo, o Avaliador de Estado do CE é executado ciclicamente. Em cada ciclo de execução, cada uma das regras *state-count-as* $r_s \in R_s$ (onde $r_s = \langle b_f, i_f, c \rangle$) é avaliada. Caso o fato bruto b_f seja consequência lógica do estado do ambiente e da instituição, é feita a avaliação do contexto c da regra. A avaliação do contexto, nesse caso, é semelhante à avaliação do fato bruto: caso o contexto seja consequência lógica do estado do ambiente e da instituição, a regra é aplicável. Sendo a regra aplicável, o CE adiciona o fato institucional f_i da regra à fila de Propriedades Objetivo ($\mathcal{T}$). Esse processamento corresponde à Definição 3.2.14. O Algoritmo 2 expressa, de forma resumida, o processamento de eventos pelo CE.

Algoritmo 2: Algoritmo para processamento de estado

```
enquanto Count-as Engine estiver sendo executado faça
    para cada r_s = ⟨b_f, i_f, c⟩ ∈ R_s faça
        se N ∪ I ⊨ b_f então
            se N ∪ I ⊨ c então
                adiciona i_f a T
            fim
        fim
    fim
fim
```

4.1.4 Carregamento de programa *Count-as* e adição de novas regras

O CE tem como função essencial processar regras *count-as*. Para carregar tais regras, foram definidas duas funções para o CE:

- `addProgram(filename)`: Carrega um programa *count-as* salvo em um arquivo (informado no parâmetro `filename`);

- `addRule(rule)`: Carrega a regra passada no parâmetro `rule`.

Para adição de uma regra única, o processo é essencialmente o seguinte: o CE recebe a regra a ser incluída através da função `addRule`. Essa regra é submetida ao *parser*, que a analisa. Se a regra estiver sintaticamente correta, o CE verifica se não possui uma regra igual em sua representação do programa *Count-as*. Se não existir, a regra é adicionada ao conjunto das regras *event-count-as* ou *state-count-as*, conforme o caso. Esse processo corresponde ao estabelecido na Definição 3.2.11. O Algoritmo 3 ilustra a função `addRule(rule)`.

Algoritmo 3: Algoritmo para a função *addRule(rule)*

```
Entrada: Regra rule
se rule é uma regra válida então
    se rule não existe no Programa Count-as então
        se r é uma regra event-count-as então
            adiciona rule ao conjunto R_e do Programa
            Count-as
        senão
            adiciona rule ao conjunto R_s do Programa
            Count-as
        fim
    senão
        erro: Regra já existente no Programa Count-As
    fim
senão
    erro de sintaxe
fim
```

Para o carregamento de um programa Count-as inteiro, foi definida a função `addProgram(fileName)`, que lê o programa contido no arquivo informado no parâmetro `filename`, o analisa e insere as regras no programa count-as armazenado pelo *Count-as Engine*. Enquanto na função `addRule` as novas regras são adicionadas ao conjunto de regras já existentes, na função `addProgram`, é criada uma nova representação para o programa count-as, correspondente ao arquivo. O processo para isso é o seguinte: o CE faz uma cópia do programa *count-as* que, ao fim do processamento, será sobrescrito. Depois disso, o *parser* analisa o arquivo. Se seu conteúdo estiver sintaticamente correto, são retornados dois conjuntos de informações contidas no arquivo: R_{temp}, com regras *count-as* e D_{temp} com expressões DKS. Para cada uma das regras: (i) o CE verifica se o programa *count-as* possui regra igual e (ii) caso não

possua, a regra é adicionada ao conjunto das regras *event-count-as* ou *state-count-as*, conforme o caso. De forma semelhante, para cada uma das expressões de conhecimento de domínio, o CE verifica se já possui armazenada uma expressão igual e, se, não possuir, insere-a ao seu conjunto. O Algoritmo 4 ilustra a função `addProgram(filename)`.

Algoritmo 4: Algoritmo para a função *addProgram(filename)*

Entrada: Arquivo fonte $filename$
Seja P o Programa Count-as atual ($\langle R_e, R_s, D \rangle$)
se *filename é um programa-fonte count-as válido* **então**
  P' = P *//cópia de P*
  o Interpretador retorna um conjunto R_{temp} de regras count-as
  para cada *regra* $r \in R_{temp}$ **faça**
    addRule(r)
  fim
  para cada *expressão* $d \in D_{temp}$ **faça**
    se $d \notin D$ **então**
      adiciona d a D ($D \cup d$)
    senão
      erro: expressão KDExpr já existente no Programa Count-As
    fim
  fim
  se *algum erro durante o processamento* **então**
    P = P' *//recupera a cópia do programa*
  fim
senão
  erro de sintaxe
fim

4.1.5 Adição de evento

Os eventos que são processados pelo processador de eventos são informados ao CE pela camada de interface e armazenados na fila de eventos $\mathcal{E}$. Considera-se que todos os eventos ocorridos no ambiente e instituição são informados e assume-se que essa informação seja de responsabilidade da camada de interface. Para possibilitar a informação

dos eventos, foi projetada no CE a função `addEvent(event)`, que é disponibiliza à camada de interface. Essa função recebe como parâmetro um evento que é, então, inserido no final da fila $\mathcal{E}$.

4.1.6 Manutenção do estado observável

Para processar regras *state-count-as*, o CE possui uma representação do estado observável do ambiente e instituição (através dos conjuntos $\mathcal{N}$ e $\mathcal{I}$, respectivamente) que se assume corresponder à realidade verificada em tais estruturas. Para isso, assume-se também que a camada de interface informe corretamente ao CE as alterações verificadas no estado observável. Para que a camada de interface possa fornecer essas informações, foram projetadas três funções para o CE:

- `addProp(prop)` para informar uma nova propriedade à representação de estado observável do CE;
- `updateProp(prop)` para atualizar uma propriedade já existente na representação de estado observável do CE;
- `removeProp(prop)` para remover uma propriedade da representação de estado observável do CE;

Para adicionar uma propriedade, o ambiente, através da camada de interface, utiliza a função `addProp(prop)` para informar a nova propriedade $p_e \in P_e$ através do parâmetro `prop`. Caso a propriedade não conste no conjunto de propriedades observáveis do ambiente ($\mathcal{N}$), é adicionada; do contrário, é ignorada. De forma semelhante, a instituição, através da camada de interface, utiliza a mesma função `addProp(prop)` para informar a nova propriedade $p_i \in P_i$ através do parâmetro `prop`. Caso a propriedade não conste no conjunto de propriedades observáveis da instituição ($\mathcal{I}$), é adicionada; do contrário, é ignorada. O Algoritmo 5 ilustra a função `addProp(prop)`.

Algoritmo 5: Algoritmo para a função addProp(prop)

Entrada: Propriedade *prop*
se $prop \in P_e$ **então**
  se $prop \notin \mathcal{N}$ **então**
    $\mathcal{N} = \mathcal{N} \cup prop$
  fim
senão
  se $prop \in P_i$ **então**
    se $prop \notin \mathcal{I}$ **então**
      $\mathcal{I} = \mathcal{I} \cup prop$
    fim
  fim
fim

Para atualizar o valor de uma propriedade observável já existente na representação mantida pelo CE, ambiente e instituição informam, através da camada de interface, a propriedade ($p_e \in P_e$ ou $p_i \in P_i$, conforme o caso), que foi alterada. Se as propriedades existirem no CE ($p_e \in \mathcal{N}$ ou $p_i \in \mathcal{I}$), a atualização é realizada. Do contrário, a solicitação de atualização é ignorada. O Algoritmo 6 ilustra a função `updateProp(prop)`.

Algoritmo 6: Algoritmo para a função updateProp(prop)

Entrada: Propriedade *prop*
Seja update(X, Y) *uma função que atualize a propriedade Y no conjunto X*
se $prop \in P_e$ **então**
  se $prop \in \mathcal{N}$ **então**
    update($\mathcal{N}$,*prop*)
  fim
senão
  se $prop \in P_i$ **então**
    se $prop \in \mathcal{I}$ **então**
      update($\mathcal{I}$,*prop*)
    fim
  fim
fim

Para remover uma propriedade observável existente na representação mantida pelo CE, ambiente e instituição informam, através da camada de interface, a propriedade ($p_e \in P_e$ ou $p_i \in P_i$, conforme o caso) a ser removida através da função `removeProp(prop)`. Se

a propriedade existir no CE ($p_e \in \mathcal{N}$ ou $p_i \in \mathcal{I}$), a remoção é realizada. Do contrário, a solicitação de remoção é ignorada. O Algoritmo 7 ilustra a função `removeProp(prop)`.

Algoritmo 7: Algoritmo para a função removeProp(prop)

```
Entrada: Propriedade prop
se prop ∈ P_e então
    se prop ∈ 𝒩 então
        𝒩 = 𝒩 \ prop
    fim
senão
    se prop ∈ P_i então
        se prop ∉ ℐ então
            ℐ = ℐ \ prop
        fim
    fim
fim
```

4.1.7 Entrega de resultados para a aplicação

O resultado do processamento das regras *count-as* é inserido na fila $\mathcal{T}$, sendo responsabilidade da camada de interface retirar esses resultados de $\mathcal{T}$. Assume-se, na arquitetura proposta, que a camada de interface retirará esses resultados continuamente, através da função `getTargetProp()`, fazendo com que o tamanho da fila $\mathcal{T}$ tenda a permanecer igual a zero. Assume-se também que todo e qualquer tratamento que seja necessário aplicar aos elementos retirados de $\mathcal{T}$ é feito pela camada de interface, restando ao CE apenas a responsabilidade de manter $\mathcal{T}$ acessível.

4.2 CAMADA DE INTERFACE

Conforme mencionado anteriormente, o modelo proposto no Capítulo 3 busca manter um bom nível de generalidade em relação a especificidades dos modelos de ambiente e instituição com que se relaciona. Por esse motivo, o modelo foi proposto sem considerar modelos de ambiente e instituição específicos. Como consequência, na arquitetura proposta neste capítulo, a camada *count-as*, composta pelo CE também foi projetada para manter generalidade em relação a infraestruturas de

ambiente e instituição e, por isso, a arquitetura não foi proposta com base em implementações específicas de ambiente e instituição.

Considerando essas premissas, admite-se que diferentes modelos e implementações de estruturas ambiental e institucional possam interagir com o CE, fornecendo informações sobre eventos e propriedades bem como, no caso específico da estrutura instucional, utilizando o resultado do processamento realizado pelo CE. Admite-se também que a existência de tais estrturas em SMA seja *anterior* à existência do CE. Ou seja, considera-se que essas estruturas podem ser componentes de um SMA sem a presença do CE, sendo este apenas um elemento que agrega a funcionalidade de criação de fatos institucionais a partir de fatos verificados no ambiente e instituição.

Por esse motivo é necessário prever, na arquitetura, uma maneira de propiciar que elementos cuja arquitetura e modelo não foram definidos considerando a existência do CE interajam com ele. Essa é a função prevista para a camada de interface: prover meios de as estruturas ambiental e institucional interagirem com o CE. Além disso, a camada de interface define acesso àquilo que chamou-se de *configurações*: elementos do CE que precisam ser acessados para ajustar o funcionamento do mesmo em conformidade com o SMA em que está inserido.

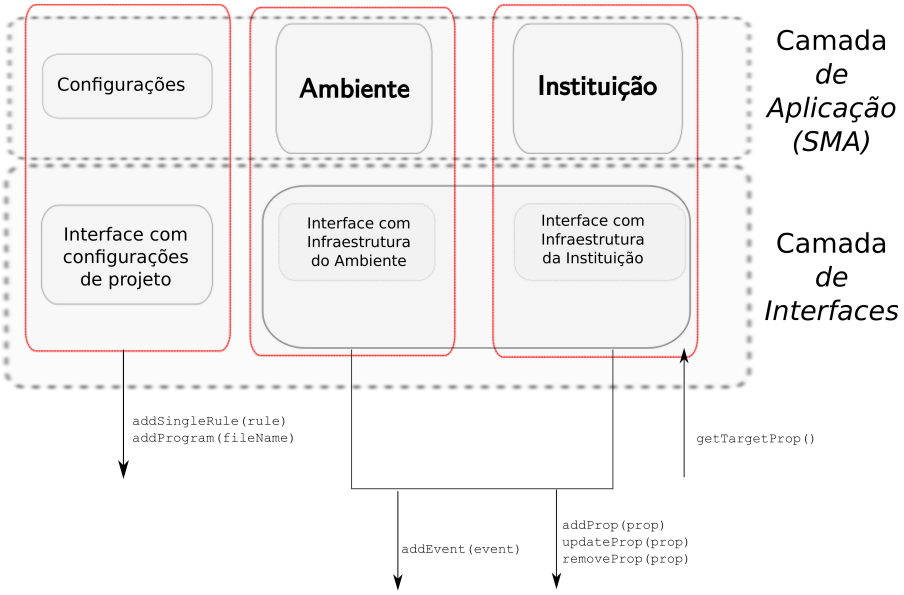


Figura 17 – Arquitetura - interfaces

A Figura 17 destaca a camada de interfaces e sua ligação com a camada de aplicação. É possível verificar na figura, através dos quadros de contorno vermelho, que os elementos da camada de interface estão ligados a elementos da camada de aplicação. Essa ligação explica-se pelo fato de as interfaces implementarem aspectos diretamente relacionados aos elementos da camada de aplicação. Apesar de a arquitetura definir uma separação entre as camadas de interface e aplicação, essa separação é conceitual. É possível, inclusive, que a interface com o CE seja incorporada na própria implementação do ambiente ou instituição.

As interfaces com a infraestrutura do ambiente e da instituição têm duas funções comuns: manter o CE atualizado com relação aos eventos e alterações de estados ocorridos na camada de aplicação. Para isso, a camada de interface pode utilizar as funções `addEvent(event)`, `addProp(prop)`, `updateProp(prop)` e `removeProp(prop)`, descritas nas seções 4.1.5 e 4.1.6. Conforme mencionado anteriormente, as interfaces devem implementar mecanismos que detectem a ocorrência de eventos nas estruturas correspondentes da camada de aplicação e informá-los ao CE. As interfaces são responsáveis, inclusive, por garantir que todos os eventos sejam informados ao CE na ordem correta e da forma correta. De forma semelhante, as interfaces devem implementar mecanismos que detectem mudanças nas propriedades observáveis e informá-las ao CE, responsabilizando-se pela consistência dessas informações.

A interface com a infraestrutura da instituição tem ainda a responsabilidade de aplicar o resultado do processamento feito pelo CE à infraestrutura institucional. Para isso, deve utilizar a função `getTargetProp()` disponibilizada pelo CE, descrita na seção 4.1.7. É importante mencionar que, conforme o modelo proposto, os elementos pertencentes ao conjunto $\mathcal{T}$ descrevem propriedades que devem passar a ser verdadeiras na instituição. É papel da interface, além de retirar as propriedades de $\mathcal{T}$, fazer com que a instituição passe a ter tais propriedades.

Uma terceira interface definida na arquitetura não tem ligação direta com as infraestruturas ambiental e institucional mas foi prevista para que seja possível interagir com o CE manipulando aspectos relacionados à sua configuração. Entre as operações que podem ser feitas por essa interface, pode-se citar o carregamento de um programa *count-as* ou a inserção de regras *count-as* e manipulação de configurações relacionadas ao funcionamento do CE. Um exemplo de configuração é a frequência com que o estado do ambiente e instituição é analisado pelo avaliador de estados. Na arquitetura proposta, esse tempo pode ser

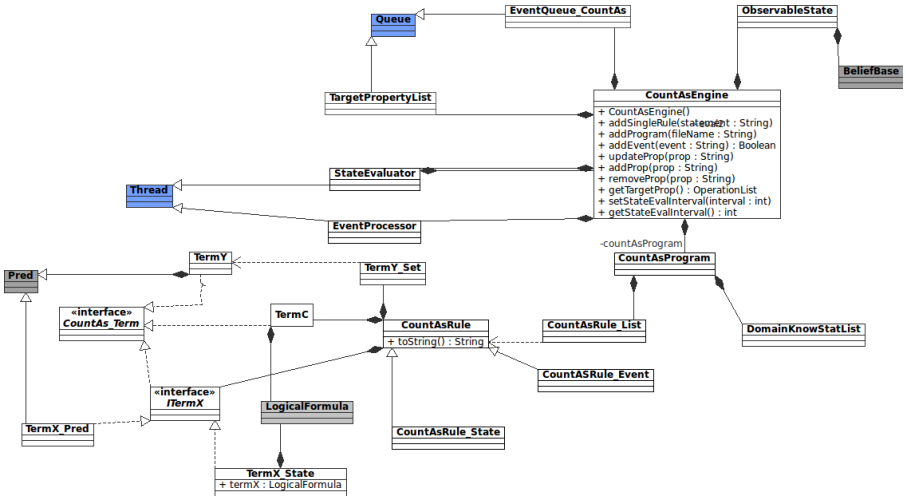


Figura 18 – Diagrama de classes do *Count-as Engine*

configurado para se adequar a aspectos como desempenho do sistema ou outros aspectos relacionados ao domínio da aplicação.

4.3 IMPLEMENTAÇÃO

Assim como o modelo proposto pode originar várias arquiteturas, sendo uma delas a apresentada neste capítulo, a arquitetura apresentada pode ser implementada de diferentes maneiras. Como parte do trabalho que gerou esta dissertação, foi realizada uma implementação com a finalidade de validar o modelo e a arquitetura propostos.

O CE foi implementado em linguagem Java e um diagrama resumido, com suas principais classes, é exibido na Figura 18. Como é possível visualizar na figura, o CE é implementado através de uma classe cujos métodos públicos correspondem às operações disponibilizadas à camada de interface. A seguir será descrita a implementação das demais estruturas definidas na arquitetura do CE, mencionando as principais classes utilizadas na implementação.

O programa *Count-as*, exibido na Figura 15, é implementado pela classe `CountAsProgram`, que é composta, por sua vez, por uma lista de regras, implementada pela classe `CountasRule_List`, e uma lista de DKS, implementada pela classe `DomainKnowStat_List`. As-

sociando a implementação ao modelo e à arquitetura, segundo a Definição 3.2.10, a classe `CountasRule_List` implementa os conjuntos R_e e R_s e a classe `DomainKnowStat_List` implementa o conjunto D.

A Fila de Eventos, implementada pela classe `EventQueue_CountAs`, implementa uma fila (interface `Queue` da linguagem Java) e corresponde ao conjunto $\mathcal{E}$ da definição 3.2.10. A classe `CountAsEngine` recebe da camada de interfaces, através do método `addEvent`, as informações sobre eventos ocorridos. À medida que esses eventos são recebidos, são inseridos em uma instância de `EventQueue_CountAs` mantida pela classe `CountAsEngine` para serem consumidos posteriormente.

O Estado Observável mantido pelo CE é implementado pela classe `ObservableState` e corresponde aos conjuntos $\mathcal{N}$ e $\mathcal{I}$ (Definição 3.2.10). A classe `ObservableState` utiliza uma instância da classe *BeliefBase*, componente da implementação da linguagem Jason. Essa opção foi feita em virtude de a classe *BeliefBase* disponibilizar funcionalidades que facilitam a verificação de consequência lógica de proposições. No caso da implementação descrita neste trabalho, o estado observável do ambiente e instituição contém as premissas a partir das quais é possível chegar a determinadas conclusões. Por exemplo, se o estado é composto pelas propriedades $professorNaSala$ e $horaAtual(10pm)$, estas são as premissas que permitem concluir que a proposição $professorNaSala$ é verdadeira.

Para implementar a fila de propriedades objetivo $\mathcal{T}$ do CE (definições 3.2.10, o 3.2.10), a classe `CountAsEngine` possui uma instância da classe `TargetPropertyList`, que também implementa uma fila (interface `Queue` da linguagem Java), em que o resultado do processamento das regras *count-as* é inserido no final da estrutura e o primeiro elemento pode ser retirado pela camada de interface.

Os eventos recebidos pelo CE são processados pelo Processador de Eventos (conforme a Figura 18). Esse elemento é implementado pela classe `EventProcessor`, que extende a classe `Thread` da linguagem Java. A classe `CountAsEngine` possui uma instância de `EventProcessor` que é executada continuamente. Durante essa execução, `EventProcessor` verifica a fila de eventos do CE, retirando sempre o seu primeiro elemento. Sobre esse elemento são aplicadas regras `event-count-as`, se houverem, conforme a Definição 3.2.13 e conforme o Algoritmo 1.

O processamento do estado do ambiente e instituição é feito pelo Avaliador de Estado (Figura 18). Esse elemento é implementado pela classe `StateEvaluator`, que estende a classe `Thread`

da linguagem Java. A classe `CountAsEngine`, assim, instancia `StateEvaluator`, que é executado continuamente enquanto a instância de `CountAsEngine` é executada. Em cada ciclo de execução, cada uma das regras *state-count-as* é avaliada em função das propriedades armazenadas em `ObservableState`, conforme a Definição 3.2.14 e conforme o Algoritmo 2.

4.3.1 *Parser*

O analisador sintático (*Parser*) é um elemento definido na arquitetura do CE com a finalidade analisar os códigos-fonte de programas *count-as* e inseri-los nas estruturas de dados adequadas. Para isso, possui duas funcionalidades essenciais: (i) interpretar um programa *count-as* completo, inserindo-o, depois de interpretado, na representação do Programa *Count-as* do CE e (ii) interpretar regras *count-as* que venham a ser informadas em tempo de execução [1]. O *parser*, ao receber uma cadeia de caracteres como entrada, analisa-a e, caso a análise seja bem sucedida, as regras *count-as* e expressões DKS são inseridas na instância de `CountAsProgram` do CE.

O *Parser*, implementado pela classe `CountAs_Parser` utiliza, na análise sintática, um analisador léxico, implementado pela classe `CountAs_Lexer`. Ambas as classes foram produzidas com a utilização do gerador de analisadores sintáticos ANTLR, que, a partir de uma gramática corretamente definida, gera o código necessário para a implementação de um *parser* (PARR; FISHER, 2011).

4.4 CONCLUSÕES

Ao apresentar o modelo do Capítulo 3, procurou-se propor um modelo genérico com relação a infraestruturas de ambiente e instituição. Assim como no caso do modelo, a arquitetura foi proposta considerando generalidade quanto a infraestruturas ambiental e institucional, sendo por isso, proposta em camadas. A camada *count-as*, composta pelo *Count-as Engine*, incorpora os elementos necessários para, efetivamente, cumprir o papel de interpretador para as regras *count-as*. Em função da independência prevista entre o CE e a camada de aplicação, definiu-se uma camada de interfaces fazer com que os elementos da camada de aplicação interajam de maneira adequada com o CE. Diferente

[1] A sintaxe da linguagem analisada pelo *Parser* é ilustrada na Figura 11.

da arquitetura “generalista” do CE, no entanto, a camada de interfaces deve ser projetada conforme as estruturas de ambiente e instituição presentes na aplicação.

No próximo capítulo, serão apresentados alguns experimentos em que o CE é incorporado SMA. Para tal, também no próximo capítulo, será apresentada a arquitetura e implementação de uma camada de interface específica para as aplicações utilizadas.

5 EXPERIMENTOS

Como uma das formas de testar e validar a arquitetura e o modelo propostos nos capítulos anteriores, optou-se por incorporar a implementação do CE em SMA cujas características estejam de acordo com aquelas para as quais o modelo foi proposto: SMA abertos e compostos por três abstrações primárias: agentes, instituição e ambiente. Para a implementação dos experimentos, optou-se em incorporar o CE ao *framework* JaCaMo (BOISSIER et al., 2011), pois este permite que um SMA seja concebido de acordo com tais características.

Este capítulo descreve dois experimentos em que a implementação do CE foi utilizada em conjunto com o *framework* JaCaMo. Para isso, inicialmente será feita uma breve descrição deste *framework* e da interface projetada e desenvolvida para ligá-lo ao CE.

5.1 O *FRAMEWORK* JACAMO

De forma resumida, a concepção de SMA, no *framework* JaCaMo, pode ser descrita como um conjunto organizado de agentes atuando em um mesmo ambiente. A abordagem do JaCaMo consiste em manter a independência entre as plataformas que implementam agentes, ambiente e organização e, mesmo mantendo essa independência, permitir que essas plataformas funcionem em conjunto, definindo dependências, conexões e mapeamentos conceituais entre as abstrações que compõem cada um dos modelos.

Com relação aos agentes, estes são implementados através da linguagem Jason, seguindo o modelo BDI. Sendo assim, um agente é composto por (i) uma série de crenças, que expressam o conhecimento do agente sobre si e sobre o mundo que o cerca, (ii) um conjunto de objetivos, que são as tarefas que um agente deseja ou precisa executar e (iii) um conjunto de planos, que definem as ações que os agentes devem executar para atingir seus objetivos.

Quanto a organização e ambiente, algumas explicações mais detalhadas serão dadas nas próximas seções. Para facilitar a compreensão, as explicações serão acompanhadas por exemplos baseados em um cenário que simula diversos agentes atuando em conjunto na redação de um artigo científico. Mais especificamente, um agente deseja redigir um artigo e outros agentes o auxiliam nessa tarefa.

5.1.1 A dimensão *ambiente* no JaCaMo

A dimensão *ambiente*, no JaCaMo, segue o modelo *Agents & Artifacts* e é implementada pelo *framework* CArtAgO, sendo concebida, como um conjunto de elementos de *software* chamados *artefatos*, sobre os quais os agentes podem atuar e ter percepções (OMICINI; RICCI; VIROLI, 2008). Para isso, os artefatos disponibilizam aos agentes um conjunto de *operações* e um conjunto de *propriedades observáveis*. As operações são processos, executados pelos artefatos, que podem ser acionados pelos agentes ou por outros artefatos. Do ponto de vista dos agentes, as operações são ações que podem ser executadas sobre o ambiente.

O estado dos artefatos é representado por *propriedades observáveis*, que podem sofrer modificações à medida que operações são realizadas sobre os artefatos. Do ponto de vista dos agentes, as propriedades observáveis são mapeadas como crenças.

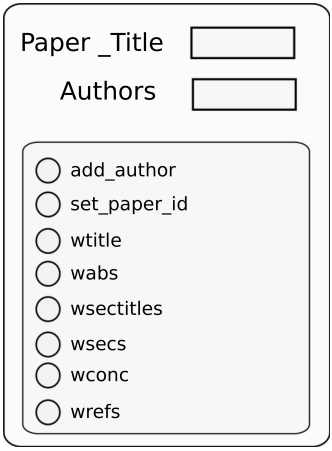


Figura 19 – Artefato *PaperArt*

No cenário proposto como exemplo, o ambiente é composto por um único artefato chamado *PaperArt*, que implementa funcionalidades que permitem que os agentes trabalhem em conjunto executando as tarefas necessárias para a redação do artigo. O artefato tem duas propriedades observáveis: `Paper_Title`, ilustrado na Figura 19, que corresponde ao título do artigo ao qual o artefato se refere e `Authors`, que corresponde ao nome dos autores do artigo. O artefato também

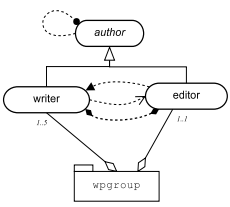
disponibiliza algumas operações aos agentes. A operação `add_author` permite que, ao ser executada, o agente se torne um dos autores do artigo. A operação `set_paper_id` permite que seja definido um identificador para o artigo· As demais operações se referem a tarefas relacionadas à redação do artigo: redação do título, do resumo, dos títulos das seções etc.

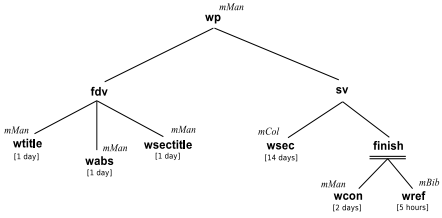
Para compreender a utilização desse artefato no cenário de exemplo, pode-se supor que os agentes irão escrever um artigo em conjunto e o artefato será utilizado para facilitar essa colaboração. Os agentes envolvidos no processo de redação desempenham suas tarefas, as quais têm uma operação correspondente no artefato. Pode-se dizer que cada operação relacionada à redação do artigo, implementada no artefato, corresponde à conclusão de uma parte da redação. Dessa forma, ao executar a operação `wtitle`, um agente informa ao artefato qual é o título do artigo. Ou, ao executar a operação `wabs`, o agente informa ao artefato o texto que compõe o resumo do artigo. O artefato *PaperArt* pode ser considerado um artefato "genérico", que não se destina a prover funcionalidades para que os agentes escrevam um determinado artigo, mas, ao contrário, diferentes instâncias do artefato podem permitir que os agentes trabalhem em diferentes artigos.

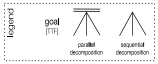
5.1.2 Organização

Conforme mencionado ao longo deste trabalho, por *instituição*, em SMA, entende-se o conjunto de mecanismos utilizados para orientar e restringir a atuação dos agentes no sistema, com o objetivo de manter a consistência do mesmo. Não foi feita, no entanto, qualquer restrição aos tipos de mecanismos que podem compor a dimensão institucional pois diferentes modelos podem utilizar diferentes mecanismos. No caso do JaCaMo, os mecanismos institucionais existentes no *framework* seguem o modelo $\mathcal{M}$oise (HÜBNER; SICHMAN; BOISSIER, 2007) e são implementados utilizando a ideia de *organização*, que será explicada a seguir. Por esse motivo, para tratar da dimensão institucional no JaCaMo, em lugar de instituição, será utilizada a palavra *organização*.

Uma organização, no $\mathcal{M}$oise, é especificada em três dimensões: estrutural, funcional e normativa. A especificação estrutural (*Structural Specification* ou SS) define a estrutura organizacional em termos de grupos de agentes, papeis e relações entre papeis e grupos. Um papel define, para os agentes que o desempenham, um padrão de comportamento que deve ser seguido. Os grupos são conjuntos de papeis que







id	condition	role	type	mission	TTF
n1		editor	per	mMan	–
n2		writer	obl	mCol	1 day
n3		writer	obl	mBib	1 day
n4	unfulfilled(n2)	editor	obl	ms	3 hours
n5	fulfilled(n3)	editor	obl	mr	3 hours
n6	#mc	editor	obl	ms	1 hour

Figura 20 – Especificações estrutural (esquerda), funcional (direita) e normativa (abaixo) (adaptado de (HÜBNER; BOISSIER; BORDINI, 2009))

têm relação entre si e que podem, em conjunto, ter relação com outros grupos. Na Figura 20 pode-se verificar a especificação estrutural de uma organização que tem por objetivo fazer com que os agentes que a compõem escrevam um artigo em conjunto. A especificação define que os papeis são reunidos em um grupo, chamado *wpgroup*. Esse grupo deve ser composto por um agente com o papel de *editor* e por, no máximo, cinco agentes com o papel de escritor (*writer*).

A especificação funcional (*Functional Specification* ou FS) define *schemas* que especificam como os grupos definidos na SS podem atingir os objetivos da organização. Os *schemas* podem ser descritos como árvores que decompõem os objetivos da organização, em que a raiz é o objetivo de todo o grupo e as folhas são os objetivos que podem ser objetivos pelos agentes individualmente. A Figura 20 exibe um *schema* para o objetivo da organização de escrever um artigo (*wp - write paper*). Para atingir esse objetivo, o *schema* define que, inicialmente, deve ser escrita uma versão prévia do artigo, através do objetivo *fdv* (*first draft version*). Considera-se que esse objetivo foi atingido quando os objetivos *wtitle* (redação do título), *wabs* (redação do resumo) e *wsectitle* (redação dos títulos das seções) forem atingidos. A especifiacação também define um tempo limite para que os objetivos sejam atingidos, sendo que para os objetivos *wtitle*, *wabs* e *wsectitle* esse tempo é de um dia. Para atingir o objetivo organizacional *wp*, depois de atingido o

objetivo *fdv*, deve ser atingido o objetivo *sv* (*submission version*). Esse objetivo é considerado atingido quando for atingido o objetivo *wsec* (redação das seções) e o objetivo *finish* que, por sua vez, será atingido quando forem atingidos os objetivos *wcon* (redação da conclusão) e *wref* (redação das referências bibliográficas). O *schema* define também que os objetivos devem ser atingidos em sequência, à exceção dos objetivos *wcon* e *ref*, que podem ser atingidos em paralelo. Além dos objetivos, a FS define *missões*. Uma missão é um conjunto de objetivos com os quais um agente pode se comprometer. Na especificação exibida na Figura 20, os objetivos estão distribuídos em três missões: *mMan* para as tarefas de gerenciamento do processo, *mCol* para a colaboração na redação do artigo e *mBib* para a redação das referências bibliográficas.

Uma terceira especificação no modelo de organização é a especifiacação normativa, que relaciona papeis a missões. As normas, no Moise, são dadas em termos de permissões e obrigações em relação ao compromisso com missões. A Figura 20 exibe a especificação normativa para o cenário utilizado como exemplo nesta seção. A norma *n1* especifica que um agente que desempenha o papel *editor* tem *permissão* para se comprometer com a missão *mMan*. A norma *n2* especifica que um agente que desempenha o papel *writer* tem *obrigação* de se comprometer com a missão *mCol*. A especificação também define que os agentes têm um dia para cumprir com essa obrigação.

A dimensão organizacional do JaCaMo, especificada pelo modelo Moise, é implementada segundo a abordagem ORA4MAS (HÜBNER et al., 2009), em que elementos de software chamados "artefatos organizacionais" implementam as funcionalidades necessárias para o funcionamento da organização. Além disso, estes artefatos permitem que agentes interajam ativamente com a organização, adotando papeis, comprometendo-se com missões etc. Os artefatos que implementam essas funcionalidades chamam-se *GroupBoard* e *SchemeBoard* (Figura 21).

5.1.2.1 Integração de agentes, organização e ambiente

Através dessa descrição, é possível ter uma ideia de como cada uma das dimensões que compõem o *framework* JaCaMo funcionam separadamente. No *framework*, no entanto, apesar da independência, essas dimensões estão conectadas aos agentes e isso tem algumas implicações conceituais e funcionais. Em nível conceitual, conforme a Figura 22, em que é possível verificar quatro mapeamentos (representados pelas

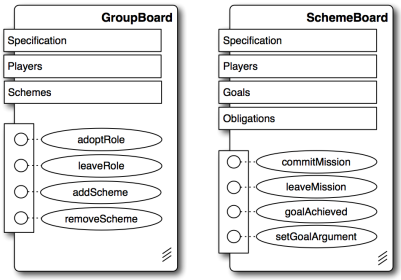


Figura 21 – Artefatos organizacionais

linhas com traço mais forte):

- As operações dos artefatos são mapeadas como operações que os agentes são capazes de realizar;
- Os sinais gerados pelos eventos ocorridos no ambiente são reconhecidos pelos agentes, que podem, de alguma forma reagir a tais ocorrências;
- Propriedades observáveis dos artefatos do ambiente são mapeadas como crenças dos agentes;
- Objetivos dos grupos se tornam objetivos dos agentes.

Do ponto de vista funcional, o JaCaMo utiliza o conceito de *workspace*, que pode ser definido como um "local" lógico em que os artefatos ambientais e organizacionais estão reunidos. Os agentes passam a ter acesso aos artefatos ao entrarem em um *workspace*, bem como os artefatos passam a ficar acessíveis uns aos outros quando estão em um mesmo *workspace*. É possível fazer uma analogia entre um *workspace* e uma sala: uma pessoa ao entrar em uma sala passa a perceber os objetos que estão dentro dela e tem a possibilidade de interagir com eles. De forma semelhante, dois objetos (artefatos) que estão na mesma sala podem passam a poder trabalhar em conjunto, como por exemplo, uma televisão e um controle remoto.

5.2 CAMADA DE INTERFACE PARA O *FRAMEWORK* JACAMO

A partir da escolha do *framework* JaCaMo, tornou-se necessário implementar a camada de interface entre o CE e as estruturas ambiental e organizacional. A Figura 23 exibe a camada de interface com

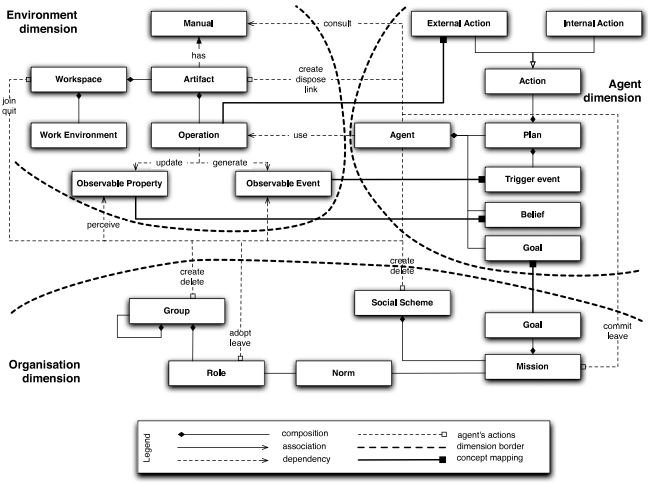


Figura 22 – Modelo de programação do JaCaMo (BOISSIER et al., 2011)

alguns aspectos específicos da implementação feita para o JaCaMo. A partir da ideia de que artefatos ambientais e organizacionais estão localizados em um mesmo *workspace*, foi projetada uma interface, chamada CE4JaCaMo que é "inserida" no *workspace* e, dessa forma, liga ambiente e instituição ao CE. Essa ligação é feita por um elemento chamado *RuleEngine*, que estende funcionalidades providas pelo CArtAgO a respeito da captação de fatos ocorridos no *workspace*. O *RuleEngine* monitora o *workspace*, capturando eventos e alterações em propriedades observáveis e repassando essas informações ao CE. Além disso, o CE4JaCaMo tem um outro elemento chamado *QueueMonitor*, que permanentemente monitora a fila de propriedades objetivo do CE, retirando os resultados do processamento das regras *count-as* e aplicando-os à estrutura institucional.

5.3 EXPERIMENTO 1 - *WRITE PAPER*

O primeiro teste foi executado sobre um SMA que implementa o cenário de agentes, em conjunto, escrevendo um artigo. Esse mesmo

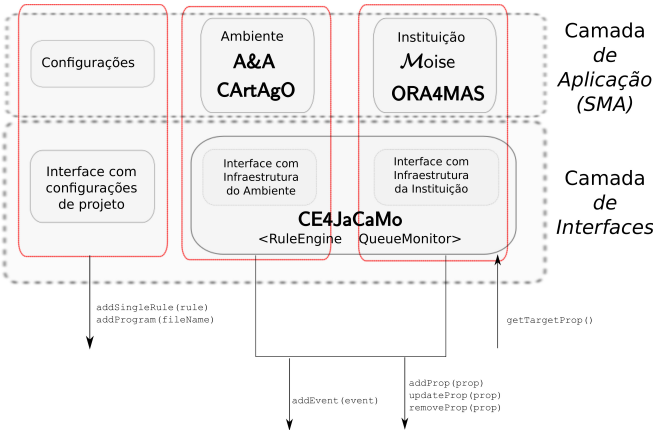


Figura 23 – Camada de interface - JaCaMo

cenário foi usado para explicar aspectos de ambiente e organização nas seções 5.1.1 e 5.1.2. A implementação original do sistema foi utilizada como exemplo em (HÜBNER; BOISSIER; BORDINI, 2009).

5.3.1 Implementação original

A implementação original não utiliza uma dimensão ambiental. Ao contrário, o sistema é composto por agentes e por uma infraeastrutura organizacional implementada com *Moise*.

De forma resumida, o cenário implementado pelo sistema é o seguinte: o agente *Bob* deseja escrever um artigo e os agentes *Carol* e *Alice* desejam colaborar com *Bob*. Sendo assim, os três agentes, em conjunto, agem para redigir o artigo. Para iniciar o trabalho em busca de seu objetivo, o agente Bob cria um grupo contendo a especificação estrutural necessária para a redação do artigo, definindo papeis a serem assumidos pelos agentes, bem como definindo relações entre esses papeis. Essa especificação foi comentada na Seção 5.1.2 e ilustrada na Figura 20. Após criar o grupo, Bob assume o papel de *editor*.

Para colaborar com Bob na redação do artigo, os agentes Carol e Alice, inicialmente, permanecem aguardando que Bob tenha a iniciativa de escrever um artigo. Quando isso acontece, passam a colaborar na redação, assumindo o papel de *escritores* no grupo criado por Bob. Com relação à implementação, os agentes Carol e Alice observam o

workspace e, quando percebem o artefato *GroupBoard* criado por Bob, inserem-se no grupo e adotam o papel de escritor .

Ao perceber que Carol e Alice adotaram o papel de escritor, Bob decide que o grupo pode começar a trabalhar. Para isso, define um fluxo de trabalho a ser seguido pelo grupo instanciando um artefato *SchemeBoard* que contém a especificação funcional (*schema*) para a redação do artigo. Essa especificação é ilustrada na Figura 20. Após instanciar o *SchemeBoard*, Bob, como editor, se compromete com a missão de gerenciar o trabalho do grupo (*mMan*).

Os agentes Alice e Carol, a partir disso, percebem que ambos são obrigados a se comprometer com a missão *mCol* e um desses agentes também precisa se comprometer com a missão *mBib*. Essas obrigações são definidas pela especificação normativa (Figura 20). Ao perceber essa obrigação, os agentes a cumprem, comprometendo-se efetivamente com as missões.

A partir do momento em que os agentes estão comprometidos com missões, estes vão percebendo, através das propriedades observáveis do artefato *SchemeBoard*, os objetivos que devem atingir. Por exemplo, ao se comprometer com a missão *mMan*, o agente Bob percebe que deve atingir o objetivo *wtitle*, que é o primeiro objetivo da missão (conforme ilustrado na Figura 20). O agente "percebe" que tem essa obrigação através da propriedade observável *Obligations* do artefato *SchemeBoard*. Apesar de a organização dizer ao agente *o que* fazer, a organização não diz *como* fazer. Ou seja, o agente delibera de forma autônoma sobre como atingir o objetivo, decidindo se quer ou não atingi-lo e executando seus próprios planos para tal [1]. Bob, então, executa seus planos para atingir o objetivo *wtitle* e, ao cumpri-lo, notifica isso à organização através da operação *goalAchieved* do artefato *SchemeBoard*. Bob então perceberá que tem uma nova obrigação: atingir o objetivo *wabs*. Reinicia-se assim o ciclo. De forma resumida, enquanto o objetivo geral da organização não for atingido, o funcionamento do sistema é basicamente esse: os agentes percebem suas obrigações a partir das propriedades observáveis do artefato *SchemeBoard*, agem autonomamente para cumprir essas obrigações e, ao cumpri-las, notificam isso à organização através da operação *goalAchieved* do artefato *SchemeBoard*.

A partir dessa descrição, é possível fazer uma análise do sistema com relação a sua implementação original. Percebe-se, pela descrição feita, que o funcionamento do sistema está fortemente apoiado na per-

[1] No caso do exemplo *Write Paper*, os agentes são projetados para sempre cumprirem os objetivos da organização.

cepção e atuação dos agentes sobre a organização. Os agentes precisam "saber" quais são os objetivos a serem cumpridos, precisam cumpri-los e, depois disso, precisam informar a organização sobre o cumprimento dos objetivos. Conforme mencionado nos capítulos anteriores, essa abordagem pode ser inapropriada em algumas situações, possibilitando a ocorrência de problemas como os exemplificados a seguir:

- Um agente pode não saber como interagir com a organização para informar o cumprimento de um objetivo. Suponha-se que Carol tenha escrito uma seção do artigo mas não tem conhecimento sobre como informar isso à instituição (considerando o sistema um SMA aberto, o projetista do agente Carol pode não ter implementado esse conhecimento). Nessa situação, mesmo tendo concluído o trabalho, a visão da organização será de que Carol ainda tem um objetivo a cumprir;

- Um agente pode agir maliciosamente e informar um objetivo cumprido sem tê-lo feito efetivamente. Suponha-se que Carol tenha um comportamento malicioso dentro do grupo e não tenha, efetivamente, intenção de colaborar com o restante dos agentes. Para isso, poderia informar à organização que uma seção do artigo foi escrita sem realmente tê-lo feito. Isso faria com que a organização, em um certo momento, considerasse o artigo como concluído sem que estivesse realmente concluído.

5.3.2 Implementação com regras *count-as*

Para adaptar o cenário à utilização de regras *Count-as* conforme o modelo proposto neste trabalho, uma primeira questão a ser tratada é a inserção da dimensão ambiental como abstração primária no sistema. No exemplo original essa dimensão não está presente e, por isso, os agentes não interagem concretamente com um elemento do ambiente. Como o modelo proposto nesse trabalho parte do princípio de que o SMA deve conter uma dimensão representando o ambiente, para a nova versão do SMA, foi necessário criá-la. A dimensão ambiental, nesse caso, foi definida como sendo composta por um único artefato chamado *PaperArt*, descrito na seção 5.1.1 e ilustrado na Figura 19. A partir da inserção deste artefato, os agentes passam a ter um elemento no ambiente sobre o qual podem atuar de forma "concreta" ao executarem suas tarefas na redação do artigo. Pode-se, por exemplo, comparar o artefato a um controlador de versão que os agentes devem utilizar para

poder redigir o artigo em equipe. Com isso, define-se, na nova versão do SMA, que, para atingir os objetivos relacionados ao artigo, os agentes precisam atuar sobre o artefato *PaperArt*.

A partir disso, foram inseridos no sistema o *Count-as Engine* e a interface CE4JaCaMo e foi escrito um programa *count-as*, ilustrado e explicado a seguir, com regras definindo fatos institucionais (que nesse caso, ocorrem na estrutura organizacional) a partir de eventos e estados do ambiente e organização.

Assim como na implementação original, na nova versão, o agente Bob inicialmente cria o artefato *GroupBoard* e, além disso, cria o artefato *PaperArt*. Na especificação da nova versão foi definido que o fato de um agente criar um artefato *GroupBoard* indica que esse agente está tendo a iniciativa de escrever um artigo, criando um grupo de trabalho para isso. Como consequência, foi definido que esse agente se torna automaticamente o editor do artigo, conforme a regra número 1 da Figura 24. Ou seja, o agente não precisa raciocinar ou agir para se tornar o editor, mas, ao contrário, isso é consequência da ação de criar um grupo.

```
/* Regra #1
   O agente que cria o grupo é o editor do artigo*/
+makeArtifact(Art,ora4mas.nopl.GroupBoard,_,_)[agent_name(Ag),artifact_name(workspace)]
 count-as
    play(Ag,editor,Art)[source(Art)].


/* Regra #2
   Se existe algum schema (artefato "SchemeBoard") instanciado, o grupo criado
   anteriormente será automaticamente responsável pelo schema*/
+makeArtifact(Art,ora4mas.nopl.SchemeBoard,_,_)[agent_name(Ag),artifact_name(workspace)]
 count-as
      responsibleFor(Art)[source(mypaper)].



/* Regra #3
   Um agente com o papel de "editor" estará automaticamente comprometido com a missão
   "mMan"  caso o schema esteja instanciado e o agente ainda não esteja comprometido
   com tal missão*/
*play(Ag,editor,mypaper)[source(mypaper)]
 count-as
    commitment(Ag,mMan,sch1)[source(sch1)]
 in
    artifact(sch1,ora4mas.nopl.SchemeBoard,sch1)[source(workspace)] &
    not(commitment(Ag,mMan,sch1)[source(sch1)]).
```

Figura 24 – Regras *count-as* - *Write Paper*

Na implementação original, após criar o grupo, Bob precisa criar um artefato *SchemeBoard* com a especificação funcional, bem como definir que o grupo criado anteriormente é responsável por atingir os

objetivos definidos no *schema*. Na especificação da nova versão, foi definido que o grupo será responsável por todo e qualquer *schema* que trate da criação de artigos. Por esse motivo, a ligação entre grupo e *schema* pode ser feita como consequência da criação do artefato *SchemeBoard* (Figura 24 - regra 2). Além disso, o agente que tem o papel de *editor* estará automaticamente comprometido com a missão *mMan* (Figura 24 - regra 3). Com isso, o agente não precisa raciocinar ou agir para fazer a ligação entre grupo e *schema* nem para compromenter-se com a missão *mMan*.

Assim como na implementação original, na nova versão os agentes Carol e Alice desejam colaborar na redação de um artigo e, para isso, procuram continuamente por um grupo de trabalho em que possam se inserir. Na implementação original, ao descobrir o grupo, esses agentes precisam ter a iniciativa de adotar o papel de escritor (*writer*). Na nova especificação, foi definido que, no momento em que um grupo é descoberto por um agente, esse agente passa a ter o papel de escritor, caso não seja o editor (Figura 25 - regra 4).

Na nova especificação foi definido também que um agente no papel de escritor estará automaticamente comprometido com a missão *mCol*, tornando-se um colaborador. Além disso, um dos agentes estará automaticamente comprometido com a missão *mBib*, tornando-se responsável pelas referêncais bibliográficas (Figura 25 - regras 5 e 6). Na implementação original, os agentes Carol e Alice precisam perceber a obrigação de se comprometer com essas missões e atuar para assumir tal compromisso. Na nova implementação, essa responsabilidade não pertence mais aos agentes.

As regras descritas anteriormente tratam de questões "estruturais" da organização (adoção de papeis, compromisso com missões etc). As regras 7 a 12 da Figura 26, por sua vez, tratam de forma mais específica, dos objetivos relacionados à redação do artigo. Resumidamente, os agentes percebem que devem atingir os objetivos *wtitle*, *wabs*, *wsectitle*, *wsec*, *wcon* e *wref*. A partir dessa percepção, executam planos para atingir seus objetivos, sendo que, independente da forma como executarão esses planos, obrigatoriamente terão que executar determinadas operações no artefato *PaperPart*. Por exemplo, para cumprir o objetivo *wsec*, correspondente à redação das seções do artigo, os agentes podem interagir entre si, consultar fontes diversas etc. Mas, em algum momento terão que executar a operação *wsec* em *PaperArt*. Na especificação da nova versão do sistema foi definido que a execução das operações *wtitle*, *wabs*, *wsectitle*, *wsec*, *wcon* e *wref* do artefato *paperArt*, dentro de determinadas condições, significaria, para organização,

```
/* Regra #4
   Se um agente "Ag" descobre um grupo de trabalho (representado pelo artefato "mypaper"
   e este agente não está desempenhando papel algum no grupo, o agente passa a ter o pap
   de escritor (writer)*/
+lookupArtifact(mypaper,_)[agent_name(Ag),artifact_name(Art)]
 count-as
   play(Ag,writer,mypaper)[source(mypaper)]
 in
   not(play(Ag,_,mypaper)[source(mypaper)]).


/* Regra #5
   Se um agente "Ag" tem o papel de escritor (writer), este agente está automaticamente
   comprometido com a missão "mCol", caso ainda não esteja e caso o schema esteja
   instanciado*/
*play(Ag,writer,mypaper)
 count-as
    commitment(Ag,mCol,Sch)
 in
   artifact(_,ora4mas.nopl.SchemeBoard,Sch) & not(commitment(Ag,mCol,Sch)).



/* Regra #6
   Um agente "Ag" está automaticamente comprometido com a missão "mBib" caso não haja
   qualquer    agente comprometido com a missão e o schema esteja instanciado */
*play(carol,writer,mypaper)
 count-as
    commitment(carol,mBib,Sch)[source(Sch)]
 in
   not(commitment(carol,mBib,Sch)[source(Sch)]) &
   artifact(_,ora4mas.nopl.SchemeBoard,Sch)[source(workspace)].
```

Figura 25 – Regras *count-as* - *Write Paper*

que os objetivos *wtitle*, *wabs*, *wsectitle*, *wsec*, *wcon* e *wref* foram atingidos. Dessa forma, os agentes precisam raciocinar apenas em trabalhar no artigo (representado pelo artefato *PaperArt*), sem se preocupar em raciocinar sobre como a organização trata a evolução do trabalho do grupo no cumprimento das diversas missões. [2]

5.3.3 Análise do experimento

Diversos pontos devem ser considerados ao analisar o experimento. Um primeiro ponto a ser abordado é a inserção da dimensão ambiental, composta pelo artefato *PaperArt*. Isso permite que, no experimento, o artigo em que os agentes trabalham tenha uma representação "concreta" e não exista somente "na mente" dos agentes.

A partir disso, é possível definir significado em nível institucional à atuação dos agentes sobre o ambiente: as operações sobre artefato *PaperArt* indicam (*count as*) o cumprimento de objetivos organizacionais. Aliado a isso, a utilização de regras *count-as* para conferir significado institucional a fatos verificados na organização permite que os agentes não precisem preocupar-se com alguns aspectos relacionados à organização e que não estão diretamente ligados a seus objetivos.

Como consequência, verifica-se uma simplificação na necessidade de raciocínio e ação dos agentes. As tabelas 2 e 3 exibem um resumo das principais ações dos agentes na execução do sistema. Conforme a Tabela 2, na implementação original, o agente Bob executa 20 ações no sistema, enquanto na implementação utilizando regras *count-as*, esse número é reduzido a 13 (redução de 35%). Para os agentes Alice e Carol, verificam-se 13 ações na implementação original e 5 ações na versão utilizando regras *count-as*, implicando em uma redução de mais de 60% (Tabela 3). A redução desse número não significa, necessariamente, melhora de desempenho na execução do sistema. Mas significa redução na quantidade de tarefas com que um agente precisa "preocupar-se". Na nova implementação, grande parte das ações dos

[2] Para manter a simplicidade do exemplo, as operações no artefato foram definidas com o mesmo nome dos objetivos da organização. Para atingir o objetivo `wsec`, a operação correspondente no artefato *paperArt* também chama-se *wsec*.Essa correspondência, no entanto, não é obrigatória. Para cumprir o mesmo objetivo *wsec*, a operação em *paperArt* poderia se chamar *write_sections*. Independente dos nomes, no entanto, os agentes precisam, de alguma forma, raciocinar sobre como executar os planos para atingir os objetivos e esse raciocínio envolve, entre outras coisas, a seleção de artefatos e operações corretas. Esse aspecto, no entanto, tem relação com a forma como os agentes planejam suas ações para satisfazer os objeivos da organização e está fora do escopo desse trabalho.

```
/* Regras #7 a #12
   A execução das operações "wtitle", "wabs", "wsectitles", "wsecs", "wrefs" e "wconc"
   por parte de um agente "Ag" sobre um artefato "paperArt1"
   faz com que um objetivo organizacional correspondente à operação
   seja atingido caso o schema esteja instanciado*/


/* Regra #7 */
+wtitle[agent_name(Ag),artifact_name(paperArt1)]
 count-as
    goalState(sch1,wtitle,Ag,Ag,satisfied)[source(sch1)]
 in
    artifact(sch1,_,_).

/* Regra #8 */
+wabs[agent_name(Ag),artifact_name(paperArt1)]
 count-as
    goalState(sch1,wabs,Ag,Ag,satisfied)[source(sch1)]
 in
    artifact(sch1,_,_).


/* Regra #9* /
+wsectitles[agent_name(Ag),artifact_name(paperArt1)]
 count-as
     goalState(sch1,wsectitles,Ag,Ag,satisfied)[source(sch1)]
 in
     artifact(sch1,_,_).


/* Regra #10 */
+wsecs[agent_name(Ag),artifact_name(paperArt1)]
 count-as
    goalState(sch1,wsecs,Ag,Ag,satisfied)[source(sch1)]
 in
    artifact(sch1,_,_).

/* Regra #11 */
+wrefs[agent_name(Ag),artifact_name(paperArt1)]
 count-as
     goalState(sch1,wrefs,Ag,Ag,satisfied)[source(sch1)]
 in
    artifact(sch1,_,_).


/* Regra #12 */
+wconc[agent_name(Ag),artifact_name(paperArt1)]
 count-as
    goalState(sch1,wconc,Ag,Ag,satisfied)[source(sch1)]
 in
    artifact(sch1,_,_).
```

Figura 26 – Regras *count-as* - *Write Paper*

agentes sobre a organização, existentes na implementação original, foi tratada por regras *count-as*. Não se pode considerar, no entanto, que foi feita uma mera "substituição" de ações dos agentes sobre a organização pela aplicação de regras *count-as*. Antes disso, pode considerar-se que, na especificação da nova versão do sistema, foi adotada uma concepção diferente, em que o estado da organização pode ser modificado por fatos brutos (eventos e estados) cuja ocorrência seja suficiente provocar tal alteração.

Tabela 2 – Ações do agente Bob

		Impl.original	Experimento
1	Criar artefato *GroupBoard*	✓	✓
2	Criar artefato *PaperArt*		✓
3	Adotar o papel de editor	✓	
4	Aguardar Carol adotar o papel de *writer*	✓	✓
5	Aguardar Alice adotar o papel de *writer*	✓	✓
6	Criar artefato *SchemeBoard*	✓	✓
7	Ligar *SchemeBoard* ao *GroupBoard*	✓	
8	Perceber a obrigação de comprometer-se com a missão *mMan*	✓	
9	Comprometer-se com missão *mMan*	✓	
10	Perceber o objetivo *wtitle*	✓	✓
11	Executar plano para *wtitle*	✓	✓
12	Informar à organização o cumprimento de *wtitle*	✓	
13	Perceber o objetivo *wabs*	✓	✓
14	Executar plano para *wabs*	✓	✓
15	Informar à organização o cumprimento de *wabs*	✓	
16	Perceber o objetivo *wsectitles*	✓	✓
17	Executar plano para *wsectitles*	✓	✓
18	Informar à organização o cumprimento de *wsectitles*	✓	
19	Perceber o objetivo *wcon*	✓	✓
20	Executar plano para *wcon*	✓	✓
21	Informar à organização o cumprimento de *wcon*	✓	
		20	13

Como decorrência disso, verifica-se uma maior simplicidade no projeto e implementação dos agentes pois se torna desnecessária parte do tratamento dado à interação entre agentes e organização e, como consequência, há uma redução de aspectos a serem considerados no projeto e na codificação dos agentes. Além disso, verifica-se uma maior

Tabela 3 – Ação dos agentes Alice e Carol

		Impl.original	Experimento
1	Inspecionar o *workspace* procurando o artefato *GroupBoard*	✓	✓
2	Adotar o papel de *writer*	✓	
3	Perceber obrigação de se comprometer com a missão *mCol*	✓	
4	Comprometer-se com missão *mCol*	✓	
5	Perceber obrigação de se comprometer com a missão *mBib*	✓	
6	Comprometer-se com missão *mBib*	✓	
7	Perceber o objetivo *wsec*	✓	✓
8	Executar plano para *wsec*	✓	✓
9	Informar à organização o cumprimento de *wsec*	✓	
10	Perceber o objetivo *wref*	✓	✓
11	Executar plano para *wref*	✓	✓
12	Informar à organização o cumprimento de *wref*	✓	
		12	5

independência entre o estado da instituição e a atuação dos agentes, uma vez que parte das alterações do estado institucional não é resultado da ação direta dos agentes, sendo apenas consequência da atuação destes sobre o ambiente. Essa independência também confere ao sistema uma redução nos riscos relacionados à autonomia dos agentes. Esses riscos foram exemplificados anteriormente, mencionando exemplos em que agentes não têm conhecimento suficiente para interagir com a instituição ou atuam de forma maliciosa.

5.4 EXPERIMENTO 2 - *BUILD A HOUSE*

O segundo experimento foi realizado a partir de um um SMA utilizado, originalmente, para exemplificar conceitos e funcionamento do *framework* JaCaMo, chamado *Build a House* (BOISSIER et al., 2011). Nesse cenário, um agente chamado *Giacomo* deseja construir uma casa. Para isso, precisa contratar empresas que possam executar tarefas relacionadas à construção, bem como fazer com que o trabalho realizado por essas empresas seja feito de forma coordenada. Para isso, a execução do sistema é dividida em duas etapas: (i) licitação, em que Giacomo procura contratar as empresas e (ii) construção, em que as diversas empresas realizam as tarefas para as quais foram contratadas.

Para cada tarefa (preparação do terreno, instalação elétrica, pintura etc) é realizada uma licitação específica, sendo que todas as licitações são realizadas simultaneamente. Para cada licitação, é instanciado um artefato *AuctionArt* (Figura 27) com propriedades observáveis indicando a tarefa para a qual a licitação se destina (`Task`), o valor máximo que o contratante deseja pagar (`Max Value`), o valor da melhor proposta recebida (`Current Bid`) e o nome do agente que fez a melhor proposta (`Current Winner`). O artefato disponibiliza uma operação `Bid` por meio da qual as empresas interessadas podem fazer suas propostas. Enquanto os artefatos são criados, diversos agentes representando empresas interessadas em prestar serviços procuram por licitações de que possam participar. Ao detectar uma licitação que seja interessante, os agentes fazem suas propostas segundo estratégias próprias de negociação. Depois de algum tempo, Giacomo decide encerrar as licitações, verificando em cada um dos artefatos as melhores propostas e contratando as empresas vencedoras.

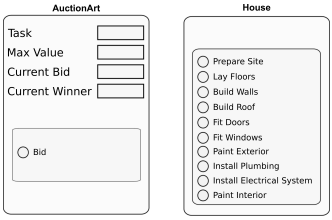
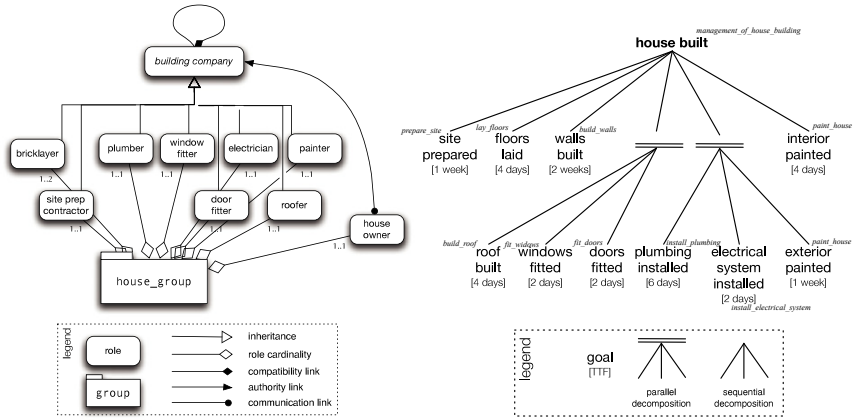


Figura 27 – Artefato *AuctionArt* (esquerda). Artefato *House* (direita)

Depois de todas as licitações encerradas, Giacomo cria uma organização virtual (através de um artefato *GroupBoard*) para auxiliar na coordenação e cooperação na execução do fluxo de trabalho (uma vez que diferentes agentes que, inicialmente, não se conhecem, devem trabalhar juntos em função de um mesmo objetivo). A especificação do grupo segue o modelo $\mathcal{M}$oise e é exibida na Figura 28, onde também é exibida a especificação funcional, que define o fluxo de trabalho para a etapa da construção.

Ao criar o grupo, o agente Giacomo adota o papel de proprietário da casa (*house owner*). Além disso, envia mensagens aos vencedores indicando que está iniciando a etapa de construção. Ao receberem as mensagens anunciando a contratação, cada um dos agentes vencedores adota um papel correspondente à licitação vencida. Por exemplo, um agente que venceu uma licitação para execução da instalação elétrica irá adotar o papel de eletricista (*eletrician*). À medida em que os agen-



norm	role	mission / goals
n1	house owner	house built
n2	site prep contractor	site prepared
n3	bricklayer	floors laid, walls built
...		

Figura 28 – *Build House* - Especificações estrutural (esquerda), funcional (direita) e normativa (abaixo) (adaptado de (BOISSIER et al., 2011))

tes adotam os papeis, a execução do sistema acontece de maneira semelhante à implementação original do SMA *Write Paper*. Inicialmente, os agentes percebem quais são as missões com que devem se comprometer e, então, assumem o compromisso. Depois disso os agentes percebem quais são seus objetivos para o cumprimento do fluxo de trabalho determinado pelo *schema*, executam planos particulares para atingir esses objetivos e, depois de atingi-los, comunicam isso à organização.

A implementação do SMA disponibiliza um artefato *House* que implementa uma interface gráfica que exibe a evolução da etapa de construção. Durante a execução de seus planos, os agentes executam operações sobre esse artefato fazendo com que o resultado da execução seja exibido na interface gráfica. Não existe, no entanto, relação entre as operações realizadas no artefato *House* e o cumprimento dos objetivos da organização.

A concepção original do SMA *Building-a-House* considera as dimensões Agente, Ambiente e Instituição, da mesma forma que o modelo proposto nesse trabalho. Percebe-se, no entanto, uma divisão bastante clara na forma como essas dimensões são utilizadas na modelagem do sistema. Na etapa de contratação, a execução do sistema é fortemente

baseada na interação entre agentes e ambiente, através da utilização, por parte dos agentes, dos artefatos de licitação. Na etapa que se segue (construção), a execução é fortemente baseada na interação entre agentes e organização. Não existe, no entanto, interação entre ambiente e organização nas duas etapas. Além disso, na implementação original, a alteração do estado da organização depende exclusivamente da atuação dos agentes, possibilitando a ocorrência de problemas semelhantes aos descritos no experimento *Write Paper*.

5.4.1 Implementação com regras *count-as*

A nova implementação, utilizando regras *count-as*, possibilita que o estado da organização se altere, não somente em função da ação dos agentes, mas também em função de eventos e estados da organização e do ambiente. Os pontos em que essa modificação foi introduzida serão descritos a seguir.

No experimento *Building-a-House* utilizou-se expressões DKS para modelar aspectos relacionados ao domínio do problema tratado. Foram definidos dois conjuntos de expressões DKS, cujo trecho de código fonte correspondente é exibido na Figura 29. O primeiro conjunto define os papeis que serão concedidos aos vencedores de cada licitação. Por exemplo, o vencedor de uma licitação chamada *auction_for_painting* assumirá, na construção da casa, o papel *painter*, definido na especificação estrutural da organização. O segundo conjunto de expressões DKS, por sua vez, define missões que serão atribuídas obrigatoriamente a agentes que estejam desempenhando determinados papeis. Por exemplo, um agente com o papel *painter* estará comprometido com a missão *paint_house*, definida na especificação funcional da organização.

Inicialmente, o evento de Giacomo criar o grupo de trabalho indica que ele é o proprietário da casa. Por isso, passa automaticamente a ter o papel de *house_owner*, sem precisar raciocinar ou atuar para efetivar a adoção do papel (Figura 30 - regra 1). Também foi definido que o grupo criado será automaticamente responsável por atingir os objetivos definidos na especificação funcional da construção da casa. Por esse motivo, grupo e *schema* são automaticamente ligados no momento em que ambos são criados (Figura 30 - regra 2). Na versão original, o agente Giacomo é o responsável por fazer essa ligação. Além disso, na nova implementação, o fato de um agente desempenhar o papel *house_owner* implica em este agente estar comprometido com a missão

```
domain_knowledge_base:
   /* role_mission(R, M): Relaciona um papel "R", definido na especificação estrutural
                          a uma missão "M", definida na especificação funcional */
   role_mission(house_owner,management_of_house_building).
   role_mission(site_prep_contractor,prepare_site).
   role_mission(bricklayer,lay_floors).
   role_mission(bricklayer,build_walls).
   role_mission(roofer,build_roof).
   role_mission(window_fitter,fit_windows).
   role_mission(door_fitter,fit_doors).
   role_mission(plumber,install_plumbing).
   role_mission(electrician,install_electrical_system).
   role_mission(painter,paint_house).

   /* auction_role(A, R): Relaciona uma licitação "A" ao papel "R" atribuído
                          ao seu vencedor */
   auction_role(auction_for_SitePreparation,  site_prep_contractor).
   auction_role(auction_for_Floors,           bricklayer).
   auction_role(auction_for_Walls,            bricklayer).
   auction_role(auction_for_Roof,             roofer).
   auction_role(auction_for_WindowsDoors,     window_fitter).
   auction_role(auction_for_WindowsDoors,     door_fitter).
   auction_role(auction_for_Plumbing,         plumber).
   auction_role(auction_for_ElectricalSystem, electrician).
   auction_role(auction_for_Painting,         painter).
```

Figura 29 – *Build House - Domain Knowledge Base*

management_of_house_building (Figura 30 - regra 3)

Na implementação original, o agente Giacomo aguarda um determinado intervalo de tempo, ao fim do qual verifica todos os artefatos de licitação, descobre seus vencedores, envia uma mensagem a esses vencedores comunicando sua contratação e estes, em função da mensagem recebida, adotam os devidos papeis. Na nova versão do SMA, os artefatos de licitação foram alterados com a inserção de uma nova operação, que permite que um agente encerre a licitação e como consequência, altere uma propriedade observável indicando que o leilão está encerrado. Dessa forma, é possível especificar que, se uma licitação estiver encerrada, seu vencedor automaticamente é contratado e adota o papel adequado no grupo de trabalho. Ou seja, na nova implementação, Giacomo precisa apenas controlar um tempo ao fim do qual encerra as licitações, não precisando verificar quem foram os vencedores, nem enviar mensagens a esses venecedores. Por fim, os agentes vencedores não precisam processar a mensagem recebida e nem agir para adotar o papel correspondente. Ao contrário, os papeis são atribuídos aos vencedores, pois, em nível institucional, é definido que o vencedor de cada leilão deve adotar um papel específico (Figura 30 - regra 4).

Na implementação original, ao adotarem os devidos papeis, os agentes percebem sua obrigação em comprometer-se com determinadas missões, conforme a especificação normativa e, mediante tal per-

```
/* Regra #1
   Um agente "Ag" que cria um artefato chamado "hsh_group" adota, automaticamente,
   o papel "house_owner" no grupo hsh_group */
+makeArtifact(hsh_group,_,_,_)[agent_name(Ag),artifact_name(workspace)]
   count-as play(Ag,house_owner,hsh_group)[source(hsh_group)].


/* Regra #2
   Se o grupo "hsh_group" estiver completo e o schema "bhsch" estiver instanciado,
   o grupo se torna responsável pelo schema */
*formationStatus(ok)[source(hsh_group)]
 count-as
    responsibleFor(bhsch)[source(hsh_group)]
 in
    not (responsibleFor(bhsch)[source(hsh_group)]).


/* Regra #3
   Um agente "Ag", desempenhando um papel "Role" no grupo "Group"
   está automaticamente comprometido com a missão "Mission"
   caso (i) ele ainda não tenha se comprometido anteriormente,
   (ii) o schema esteja instanciado e (iii) exista, na base de conhecimento de domínio,
   alguma relação entre o papel "Role" e a missão "Mission" */
*play(Ag,Role,Group)
 count-as
    commitment(Ag,Mission,bhsch)[source(bhsch)]
 in
    not(commitment(Ag,management_of_house_building,bhsch)[source(bhsch)]) &
    artifact(_,ora4mas.nopl.SchemeBoard,bhsch)[source(workspace)] &
    role_mission(Role, Mission).


/* Regra #4
   Se uma licitação, representada por um artefato "Art" estiver encerrada,
   o agente vencedor ("Winner") passa a ter um papel "Role", correspondente à
   licitação em questão, caso ainda não tenha adotado tal papel */
*auctionStatus(false)[source(Art)]
 count-as
   play(Winner,Role,hsh_group)[source(hsh_group)]
 in
   currentWinner(Winner)[source(Art)] & not(Winner==no_winner) &
   auction_role(Art,Role) & not(play(Winner,Role,_)[source(hsh_group)]).
```

Figura 30 – Regras *count-as* - *Build a House*

cepção, cumprem a obrigação, comprometendo-se com as missões. Na implementação realizada para este experimento os agentes não necessitam perceber tal obrigação e nem atuar sobre a organização para comprometer-se com missões. Ao contrário, a adoção de uma missão é considerada como consequência da adoção de um papel (Figura 30 - regra 3).

Depois disso inicia-se o processo de construção da casa propriamente dito. Na implementação original, os agentes atuam sobre um artefato *House* para exibir a progressão da construção da casa em uma interface gráfica. Essa atuação, no entanto, não possui qualquer ligação com o estado dos objetivos da organização. Caso os agentes não atuem sobre o artefato *House* mas, ainda assim, interajam com a organização, os objetivos organizacionais serão cumpridos. O contrário também é verdadeiro: mesmo que os agentes não interajam com a organização, caso atuem sobre o artefato artefato *House*, a interface pode exibir uma evolução que não é, necessariamente, correspondente ao estado da organização. No experimento, o artefato *House* passou a ser utilizado como o elemento que representa efetivamente a atuação dos agentes no cumprimento e objetivos relacionados à construção da casa. Dessa forma, cada operação executada no artefato, além de provocar a atualização da interface gráfica, tem um significado em nível organizacional. Por exemplo, ao executar a operação `fitWindows`, um agente fará com que a interface gráfica sinalize que as janelas foram concluídas e, na organização, o objetivo *fit_windows* será considerado concluído. A partir dessa concepção é possível definir, através de regras *count-as*, que as ações executadas sobre esse artefato signicam, em nível organizacional, que certos objetivos foram atingidos, conforme a Figura 31, nas regras 5 a 14.

5.4.2 Análise do experimento

Uma primeiro ponto a ser considerado no experimento é a interação entre as dimensões ambiental e organizacional. Enquanto a implementação original do exemplo *Write Paper* não contém uma dimensão ambiental, a implementação original do exemplo *Building a House* contém. Porém não há interação entre tal dimensão e a dimensão organizacional. A introdução dessa interação permitiu que, mesmo mantendo a implementação original, boa parte das mudanças de estado da organização, que antes eram resultado direto da atuação dos agentes, passasse a ser resultado de eventos e estados do ambiente

```
/* Regras #5 a #14
   A ocorrência dos eventos "prepareSite", "layFloors"... siginfica
   que alguns objetivos da organização foram atingidos.*/

/* Regra #5 */
+prepareSite[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,site_prepared,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #6 */
+layFloors[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,floors_laid,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #7 */
+buildWalls[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,walls_built,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #8 */
+buildRoof[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,roof_built,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #9 */
+fitWindows[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,windows_fitted,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #10 */
+fitDoors[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,doors_fitted,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #11 */
+installPlumbing[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,plumbing_installed,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #12 */
+installElectricalSystem[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,electrical_system_installed,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #13 */
+paintExterior[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,exterior_painted,Ag,Ag,satisfied)[source(bhsch)].

/* Regra #14 */
+paintInterior[agent_name(Ag),artifact_name(housegui)]
 count-as goalState(bhsch,interior_painted,Ag,Ag,satisfied)[source(bhsch)].
```

Figura 31 – Regras *count-as* - *Build a House*

e da organização.

Essa mudança produziu um outro resultado, semelhante ao verificado no experimento *Write Paper*: a simplificação da necessidade de ação dos agentes. A Tabela 4 exibe uma comparação entre as principais ações executadas pelo agente Giacomo na implementação original e no experimento. Para fazer essa comparação, considera-se que, no cenário proposto, todas as ações executadas pelo agente têm o efeito desejado e não precisam ser repetidas. A coluna *execuções* indica, neste cenário, quantas vezes cada ação precisa ser executada. A utilização dessa coluna deve-se ao fato de algumas ações serem executadas mais de uma vez em função de o agente manipular mais de uma licitação. As ações relacionadas à criação de artefatos ou ao encerramento de licitações, por exemplo, são executadas oito vezes, pois este é o número total de licitações realizadas. Conforme indicado na Tabela 4, na implementação original, o agente Giacomo executa, basicamente, 46 ações, enquanto no experimento esse número é reduzido em aproximadamete 58%, para 19 ações.

Tabela 4 – Ações do agente Giacomo

		Impl.original	Experimento	Execuções
1	Criar artefato *GroupBoard*	✓	✓	1
2	Adotar o papel de *house owner*	✓		1
3	Comprometer-se com a missão *management_of_house_building*	✓		1
4	Criar artefato *SchemeBoard*	✓	✓	1
5	Ligar o *SchemeBoard* ao *GroupBoard*	✓		8
6	Criar artefatos *AuctionArt*	✓	✓	8
7	Controlar o tempo das licitações	✓	✓	1
8	Encerrar licitações	✓	✓	8
9	Verificar vencedores	✓		8
10	Contratar vencedores	✓		8
11	Verificar se formação do grupo está completa	✓		1
		46	19	

Quanto aos agentes que representam as empresas que realizam as tarefas necesárias à construção da casa, também é possível verificar essa redução na quantidade de ações que os agentes executam. Como cada agente tem um comportamento diferente, a quantidade de ações que cada um executa também varia. Para fazer a mesma comparação feita para o agente Giacomo, foi selecionado um agente chamado *CompanyA*.

Neste caso, o número de ações é reduzido em 50%, de oito para quatro (Tabela 5).

Tabela 5 – Ações do agente CompanyA

		Impl.original	Experimento
1	Procurar pelo artefato *GroupBoard*	✓	✓
2	Procurar por artefatos *AuctionArt*	✓	✓
3	Fazer oferta nas licitações *AuctionArt*	✓	✓
4	Processar contratação	✓	
5	Adotar papel correspondente à licitação vencida	✓	
6	Comprometer-se com a missão correspondente ao papel adotado	✓	
7	Executar tarefas relacionadas à missão	✓	✓
8	Informar à organização o cumprimento de um objetivo	✓	
		8	4

Um último aspecto a ser observado no experimento *Build a House* é a utilização de expressões DKS para modelar o conhecimento relacionado ao domínio da aplicação que é de interesse da instituição. No caso do experimento descrito, foi possível definir correspondência entre licitações e papeis e entre papeis e missões. Na implementação original, os agentes precisam possuir esse conhecimento, enquanto no experimento, os agentes podem atuar sem possui-lo e, ainda assim, o sistema atinge os objetivos esperados.

5.5 ANÁLISE GERAL DOS EXPERIMENTOS

Além de análises relacionadas a cada um dos experimentos em particular, é possível fazer algumas análises provenientes de conclusões mais amplas, decorrentes dos dois experimentos. Essas conclusões, em alguns pontos, têm ligação direta com a forma como os elementos componentes interagem no JaCaMo.

Para fazer essa análise, torna-se necessário retomar o conceito de *instituição* como o conjunto de mecanismos utilizados para manutenção do funcionamento do sistema em conformidade com os obetivos para os quais foi projetado. Segundo essa abordagem, no JaCaMo, a dimensão institucional é formada, originalmente, pela infraeastrutura organizacional. A instituição, no entanto, pode ser vista como algo mais

abstrato e abrangente que é servido pelos mecanismos da estrutura organizacional. Utilizando-se o exemplo do experimento *Build a House*, por exemplo, pode-se dizer que o objetivo da instituição é construir uma casa e que a *organização* serve à instituição em funções como, por exemplo, garantir que os agentes atuem em conformidade com os papeis previamente definidos, fazer com que os agentes comprometam-se com missões etc. Um outro exemplo de mecanismo de caráter institucional (que não está presente nos experimentos) são os mecanismos de punição. Um mecanismo de punição pode ser visto como uma forma de manter o funcionamento do sistema de acordo com o projetado, podendo, assim, ser classificado como uma ferramenta de caráter institucional. Caso fosse incorporado ao JaCaMo, um mecanismo dessa natureza, mesmo que não fosse membro da organização, *serviria* à instituição em seu objetivo.

A partir dessa análise, pode-se concluir que, nos experimentos realizados, a dimensão institucional provida pelo JaCaMo, composta pela infraeastrutura organizacional, foi enriquecida com a incorporação do CE. Justifica-se essa afirmação pelo fato, de, nos experimentos, o CE contribuir para fazer com que o funcionamento do sistema aconteça de acordo com o projetado inicialmente. No exemplo *Build a House*, por exemplo, o cumprimento dos objetivos da organização se dá através da execução de operações sobre o artefato *House*. Caso o CE seja desativado, mesmo que os agentes atuem sobre o artefato, os objetivos da organização não serão considerados como atingidos. Sendo assim, pode-se considerar que o CE é um mecanismo institucional que dá *significado institucional a eventos e estados verificados no ambiente e nos demais mecanismos institucionais.*

Além disso, o conhecimento modelado pelas sentenças DKS pode ser considerado como um conhecimento da *instituição*. Para tratar dessa questão, pode-se utilizar como exemplo a relação entre uma licitação e o papel atribuído ao seu vencedor. Do lado dos agentes, um agente deseja atuar como *pintor* na construção da casa e, para isso, participa da licitação adequada, identificada pelo nome *auction_for_Painting*. Esse agente, no entanto, não precisa saber que o nome do papel correspondente na organização é *painter*. A estrutura organizacional poderia, por exemplo, ter sua especificação estrutural modificada, fazendo com que tal papel passasse a ser denominado *pintor* em vez de *painter*. Nesse caso, a mudança poderia ser descrita nas sentenças DKS sem que os agentes precisassem adquirir novos conhecimentos para atuar no sistema. Do ponto de vista do ambiente, o artefato que implementa a licitação *auction_for_Painting* é exatamente

igual aos artefatos que implementam as demais licitações e não tem em si mesmo qualquer informação que determine que seu vencedor deve passar a desempenhar o papel *painter*. A organização, por sua vez, possui toda a definição de papeis mas não tem condições, por si mesma, de atribuir os papeis aos agentes. Sendo assim, o conhecimento a respeito da relação entre licitações e papeis não é determinado nos agentes, no ambiente ou na organização. Mas, ainda assim, é um conhecimento necessário para que o sistema atinja os objetivos para os quais foi criado. Sendo assim, avalia-se que é um conhecimento da *instituição*.

Através dos experimentos, foi possível verificar uma ampliação no papel desempenhado pelas dimensões ambiental e organizacional com relação ao disponibilizado originalmente no *framework* JaCaMo. Originalmente, essas dimensões, ainda que integradas em um mesmo *framework*, têm funções claramente separadas: a organização modela e implementa mecanismos que auxiliem no cumprimento dos objetivos da sociedade de agentes, enquanto o ambiente modela e implementa elementos que servem aos agentes de diversas formas, *inclusive* para a execução de tarefas relacionadas ao cumprimento dos objetivos da sociedade. A partir dos experimentos, com a utilização de regras *count-as*, ambiente e organização passam, potencialmente, contribuir para alteração de estados da instituição e, consequentemente, para a satisfação dos objetivos da sociedade de agentes. É importante observar, no entanto, que ambiente e instituição continuam sendo dimensões que abrigam elementos sem qualquer autonomia, diferente dos agentes. Por isso, em si mesmo, não têm qualquer poder para fazer com que o sistema atinja os objetivos para que foi projetado. Além disso, os elementos que compõem o ambiente e organização no JaCaMo não contêm em si qualquer elemento que possa, através de suas características provocar mudanças no estado da instituição. Isso quer dizer que, no ambiente, os artefatos, com suas operações, propriedades observáveis, sinais e outros componentes, não têm, em si, elementos que possam fazer o estado da instituição ser alterado. O mesmo acontece com a organização: seus grupos, papeis, missões etc, também não são capazes de produzir tal efeito. Ao contrário, esses elementos, originalmente, podem ser manipulados (originalmente, pelos agentes) para isso. As regras *count-as*, no entanto, conferem ao ambiente e organização a capacidade de alterar o estado da instituição.

Um outro aspecto observado em decorrência dos experimentos foi a redução na necessidade e raciocínio dos agentes. Conforme mencionado anteriormente, essa redução não significa, necessariamente, melhoria no desempenho do sistema. A vantagem dessa redução também

não significa, necessariamente, simplificação na codificação dos agentes (apesar de isso ter sido observado nos experimentos). Antes disso, essa redução significa uma mudança na *concepção* da atuação dos agentes no sistema: os agentes atuam no que, de fato, lhes "interessa", sem precisar raciocinar ou atuar sobre a organização.

6 CONCLUSÕES

Em SMA que consideram agentes, ambiente e instituição como abastrações primárias, as interações entre agentes e ambiente e agentes e instituição são questões relativamente bem resolvidas. De maneira diferente, interações entre ambiente e instituição apresentam pontos ainda em aberto na área de SMA. Por esse motivo, este trabalho concentrou-se, inicialmente, em interações entre as dimensões ambiental e institucional, com preocupação especial na modelagem de interações que, a partir de fatos ocorridos no ambiente, provocam alterações no estado da instituição. O objetivo inicialmente estabelecido para o trabalho, descrito no Capítulo 1, foi propor um modelo de interação entre as dimensões institucional e ambiental em SMA, de modo a permitir que o estado da dimensão institucional seja modificado em função de fatos ocorridos na dimensão ambiental.

A partir do estudo inicial realizado para obtenção de um referencial teórico, chegou-se a uma primeira conclusão: *da mesma forma que fatos verificados no ambiente podem provocar alterações no estado da instituição, fatos verificados na própria instituição também podem provocar tal alteração.* Essa conclusão tem por base a teoria de John Searle e também alguns modelos de SMA analisados. A partir disso, verificou-se a possibilidade de abrigar essa questão no modelo proposto, aumentando assim a abrangência do mesmo. Sendo assim, passou-se a considerar que os *fatos brutos* que determinam a criação de *fatos institucionais*, podem ser provenientes tanto do ambiente quanto da instituição.

A partir disso, foi possível trabalhar de forma a atingir o primeiro objetivo específico deste trabalho, que é analisar de que forma a dinâmica de funcionamento do ambiente pode alterar o estado da instituição. Junto a isso, analisou-se de que forma a dinâmica de funcionamento da instituição pode alterar o estado da própria instituição. Antes de qualquer coisa, foi estabelecido que o estado do ambiente e da instituição é caracterizado pelo conjunto de propriedades que compõem essas estruturas. Assim, passou-se a analisar como o funcionamento do ambiente e da instituição provocam alterações nas propriedades dos elementos da instituição. Alguns dos modelos analisados consideram que essa alteração acontece em função de eventos enquanto outros modelos consideram que isso acontece em função de determinados estados que passam a ser verdadeiros. Através de um estudo sobre essas diferentes abordagens, verificou-se que cada uma delas apresenta vantagens

em determinadas situações. A partir da ideia de propor um modelo aplicável a sistemas abertos, em que não há garantia de conhecer-se integralmente as dimensões ambiental e institucional, concluiu-se que, em um modelo com generalidade suficiente, é conveniente que tanto eventos quanto estados verificados no ambiente e na instituição sejam capazes de provocar alterações no estado da instituição.

A conclusão a respeito de como a dinâmica das infraestruturas ambiental e institucional pode determinar mudanças de estado na instituição tem relação direta com o modelo proposto. No modelo, assume-se que existe a possibilidade de não se conhecer totalmente os eventos e estados possíveis de serem verificados no sistema. Por esse motivo, considera-se que os eventos e estados *observáveis* do ambiente e instituição provocam alteração em estados na instituição. Essa alteração pode ser modelada através de regras, que, no modelo, foram chamadas *regras count-as*. Juntamente com regras *count-as*, um outro elemento que se mostrou necessário foram as *Domain Knowledge Statements* (DKS), que modelam aspectos cuja semântica não é de domínio específico do ambiente nem da instituição. No modelo, esses elementos foram agregados em uma estrutura chamada *Count-as engine*, que fornece dinâmica a tais elementos.

Para validar o modelo, foi definida uma linguagem de programação para descrever regras *count-as* e DKS e uma arquitetura para viabilizar o funcionamento do modelo. Além disso, a arquitetura proposta foi implementada em *software* e utilizada em dois experimentos. Com a síntese do referencial teórico, sistematização do modelo desenvolvimento de arquitetura, implementação e experimentos, é possível extrair algumas conclusões.

Uma delas é a aproximação do modelo de sociedade de agentes à realidade verificada nas sociedades humanas. Nas sociedades humanas, em muitas situações, as instituições relacionam-se com o ambiente mas não são, necessariamente, "determinadas" pelo ambiente. Retomando um exemplo usado em diversos pontos deste trabalho, uma universidade pode precisar de salas de aula para existir, mas não são as salas de aula que constituem a essência da universidade. Uma universidade pode, por exemplo, mudar de endereço, mudando, consequentemente, suas salas e demais instalações, e, ainda assim, professores, alunos e funcionários continuarão a exercer suas funções, alunos que obtiverem um certificado serão considerados graduados, etc. De forma semelhante, na realidade humana, o ambiente, apesar de influenciado, não é determinado pelas instituições. Novamente utilizando o exemplo de uma universidade, caso a univesidade mude de endereço, a antiga sala de

aula continuará sendo uma sala, e poderá servir a uma outra instituição qualquer. Em SMA, segundo a concepção adotada neste trabalho, instituição e ambiente são elementos conceitualmente separados. A manutenção dessa separação, no entanto, pode implicar em ausência de interação entre tais dimensões, pois não há representação de elementos de uma dimensão presente em outra dimensão. Searle observou essa separação nas sociedades humanas e constatou que a ligação dessas duas dimensões é feita através de *constitutive rules*. Em SMA, a partir do modelo proposto e das validações aplicadas, foi possível constatar que a utilização de *constitutive rules* é capaz de ligar essas dimensões conceitualmente separadas.

Outro aspecto em que as sociedades de agentes passam a se assemelhar mais às sociedades humanas é a possibilidade de os agentes interagirem apenas com o ambiente sem conhecer com clareza aspectos relacionados à instituição. Considerando o ambiente como o meio que contém os elementos sobre os quais os agentes podem atuar, a evolução da instituição pode ser algo "transparente" aos agentes, se assim for conveniente. Um exemplo desse aspecto pode ser visto no experimento com o SMA *Write Paper* (seção 5.3), em que os agentes atuam sobre o artigo em que estão trabalhando enquanto a instituição tem seu estado alterado sem necessitar a atuação direta dos mesmos. As sociedades humanas também apresentam essa característica. Ao utilizar dinheiro, por exemplo, as pessoas não precisam ter noção de todo o aparato institucional que suporta o sistema monetário vigente.

Um impacto da utilização de regras *count-as*, segundo o modelo proposto neste trabalho, é a contribuição à *persistência* de um SMA. Segundo (COSTA, 2011), um SMA persistente "é aquele cujas estruturas e funções, em cada instante, tem existência e funcionamento independente do particular conjunto de agentes que habitam a sociedade naquele instante, de modo que tais estruturas e funções são capazes de persistirem no tempo, independentemente da entrada e saída de agentes do sistema". A partir desse trabalho, propõe-se uma definição um pouco diferente, tratando SMA persistente como *aquele cujas funções, em cada instante, tem existência e funcionamento independente do conjunto de agentes e das estruturas que compõem o ambiente e instituição que compõem o sistema, de modo que as funções do sistema sejam capazes de persistir no tempo, independente de alterações no conjunto de agentes ou nos elementos componentes de ambiente e instituição.* Essa definição tem por base a ideia de que, através de regras *count-as* segundo o modelo proposto neste trabalho, componentes da instituição e do ambiente podem ser modificados e, ainda assim, a dinâmica da

instituição pode ser garantida através de regras *count-as*. Como exemplo, suponha-se uma universidade em que a entrega de um formulário corretamente preenchido *count-as* um aluno estar matriculado em uma disciplina. Suponha-se, no entanto, que, em um dado momento, seja disponibilizada a alternativa de realização de matrícula através da Internet. Nesse caso, a utilização correta de um sistema de matrícula *on line count-as* um aluno estar matriculado em uma disciplina. Esse cenário pode ainda sofrer modificações, definindo-se, por exemplo, que apenas matrículas feitas através da Internet são válidas. Nesse exemplo, verifica-se alterações nos elementos que compõem ambiente e instituição, porém a função específica da sociedade que trata da efetivação da matrícula do aluno permanece inalterado.

Uma última conclusão decorrente deste trabalho é o fato de o modelo conferir ao SMA a característica do poder institucional (*empowerment*) (JONES; SERGOT, 1996), que trata do poder, concedido pela instituição ao agente, de que determinadas ações que este venha a executar tenham efeito na instituição. Pode-se supor que, no experimento *Build a House*, para participar das licitações, os agentes precisem ter o papel *Company*. Suponha-se, assim, que um agente com esse papel, ao executar a `bid` do artefato `AuctionArt`, faça uma oferta válida, enquanto a execução da operação por parte de um agente que não tenha o papel não tenha qualquer significado. Nesse caso, o agente que tem o papel *Company* tem o poder institucional de fazer com que a operação *bid* seja, na instituição, uma oferta válida em uma licitação.

6.1 TRABALHOS FUTUROS

Durante a pesquisa descrita neste trabalho, algumas questões foram levantadas mas que, devido à delimitação de escopo e tempo, não foram aprofundadas. Algumas delas são citadas nessa seção.

- Análise de desempenho do CE e JaCaMo. Nos experimentos realizados, o objetivo foi analisar o impacto resultante na concepção de um SMA, sendo a implementação do CE, do *framework* e do SMA um *meio* para atingir tal objetivo. Uma outra análise possível de ser realizada relaciona-se ao desempenho dos sistemas, detectando melhorias ou pontos passíveis de modificação no CE e na interface CE4JaCaMo.

- Alterações no estado institucional em função de interações entre os agentes. Neste trabalho, o foco manteve-se na mudança de es-

tado da instituição em função de fatos verificados na instituição e no ambiente. Na teoria de Searle verifica-se que interações entre pessoas também podem ter significado institucional. Por exemplo, o envio de uma mensagem entre líder e subordinado pode significar, institucionalmente, uma ordem.

- Incorporação do CE em SMAs que utilizem outras estruturas para ambiente e instituição.

- Análise do raciocínio dos agentes sobre regras *count-as*. Pelo fato de as regras *count-as* definirem de que forma as instituições podem ser levadas a determinados estados, um assunto passível de estudo seria a utilização dessas regras, por parte dos agentes, para que esses "descubram" as ações que precisam executar para atingir determinados objetivos relacionados à instituição.

- Possibilidade de criação de regras *count-as* pelos agentes. Este trabalho concentrou-se em definir um modelo para tratar regras *count-as* presentes em um SMA sem qualquer preocupação sobre *como* essas regras são definidas. Nos experimentos realizados essas regras foram definidas por ocasião do projeto do sistema. Nas sociedades humanas, de acordo com a teoria de John Searle, essas regras, sejam elas formais ou informais, são criadas pelos próprios membros da sociedade em função dos objetivos das pessoas e da própria sociedade. Verifica-se assim, como uma possibilidade de trabalho futuro, um estudo sobre como os próprios agentes membros de um SMA podem criar regras *count-as*, à semelhança do que acontece com as sociedades humanas.

REFERÊNCIAS

ALDEWERELD, H. et al. Engineering social reality with inheritance relations. In: *Proceedings of the 10th International Workshop on Engineering Societies in the Agents World X*. Berlin, Heidelberg: Springer-Verlag, 2009. (ESAW 09), p. 116–131.

ALDEWERELD, H. et al. Making norms concrete. In: HOEK, W. van der et al. (Ed.). *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. Toronto, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2010. p. 807–814.

ALSTON, L. J. Institutions, institutional change and economic performance - douglass c. north. *Journal of Economic Behavior and Organization*, Cambridge University Press, v. 18, n. 1, p. 142–144, 1992.

ARTIKIS, A.; PITT, J.; SERGOT, M. Animated specifications of computational societies. In: *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 3 - AAMAS '02*. New York, New York, USA: ACM Press, 2002. p. 1053–1062.

ARTIKIS, A.; SERGOT, M.; PITT, J. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic*, ACM, v. 10, n. 1, p. 1–42, jan. 2009. ISSN 15293785.

ARTIKIS, A.; SERGOT, M. J. Executable specification of open multi-agent systems. *Logic Journal of the IGPL*, v. 18, n. 1, p. 31–65, 2010.

BÁEZ-BARRANCO, J.; STRATULAT, T.; FERBER, J. A unified model for physical and social environments. In: *Proceedings of the 3rd international conference on Environments for multi-agent systems III*. Berlin, Heidelberg: Springer-Verlag, 2007. p. 41–50.

BITTENCOURT, G. *Inteligência Artificial: Ferramentas e Teorias*. 2 edição. ed. [S.l.]: Editora da UFSC, 2001.

BOISSIER, O. et al. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, n. 0, 2011.

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. J. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. [S.l.]: J. Wiley, 2007.

CAMPOS, J. et al. Formalising situatedness and adaptation in electronic institutions. In: HÜBNER, J. et al. (Ed.). *Coordination, Organizations, Institutions and Norms in Agent Systems IV*. [S.l.]: Springer Berlin / Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5428). p. 126–139.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to Discrete Event Systems*. [S.l.]: Springer, 2008. 770 p. ISSN 01966553. ISBN 9780387333328.

CASTELFRANCHI, C. Engineering social order. *Order A Journal On The Theory Of Ordered Sets And Its Applications*, Springer- Verlag, v. 1972, p. 1–18, 2000.

COSTA, A. C. da R. O nível cultural das sociedades de agentes. In: GIMÉNEZ-LUGO, G. A.; HÜBNER, J. F.; TACLA, C. A. (Ed.). *Anais do V Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCacoes (V WESAAC)*. [S.l.: s.n.], 2011. p. 3–36.

CURY, J. E. R. Teoria de Controle Supervisório de Sistemas a Eventos Discretos. In: *V Simpósio Brasileiro de Automação Inteligente*. [S.l.: s.n.], 2001.

DASTANI, M. et al. Knowledge representation for agents and multi-agent systems. In: MEYER, J.-J. C.; BROERSEN, J. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2009. cap. Normative Multi-agent Programs and Their Logics, p. 16–31.

DASTANI, M.; TINNEMEIER, N.; MEYER, J.-J. A programming language for normative multi-agent systems. In: _____. *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. Hershey, PA, USA: Information Science Reference, 2009. cap. XVI, p. 397–417.

DEMAZEAU, Y.; COSTA, A. C. R. Populations and Organizations in Open Multi-agent Systems. In: *Proceedings of the National Symposium on Parallel amd Distributed Artificial Intelligence (PDAI'96)*. Hyderabad, India: [s.n.], 1996.

ESTEVA, M. *Electronic institutions. from specification to development*. Tese (Doutorado) — Universitat Politecnica de Catalunya, 2003.

ESTEVA, M. et al. On the formal specification of electronic institutions. In: *Agent-mediated Electronic Commerce: the European AgentLink Perspective - Lecture Notes in Artificial Intelligence 1991*. [S.l.]: Springer-Verlag, 2001. p. 126–147.

ESTEVA, M. et al. Ameli: An agent-based middleware for electronic institutions. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, IEEE Computer Society, p. 236–243, 2004.

FERBER, J.; MICHEL, F.; BÁEZ-BARRANCO, J.-A. Agre: Integrating environments with organizations. In: *E4MAS*. [S.l.: s.n.], 2004. p. 48–56.

HÜBNER, J.; BOISSIER, O.; BORDINI, R. Normative programming for organisation management infrastructures. In: *MALLOW Workshop on Coordination, Organization, Institutions and Norms in Agent Systems in Online Communities (COIN-MALLOW 2009)*. [S.l.]: Citeseer, 2009.

HÜBNER, J. F. et al. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems*, v. 20, n. 3, p. 369–400, abr. 2009. ISSN 1387-2532.

HÜBNER, J. F.; SICHMAN, J. S.; BOISSIER, O. Developing organised multiagent systems using the moise. *IJAOSE*, v. 1, n. 3/4, p. 370–395, 2007.

JONES, A.; SERGOT, M. A formal characterisation of institutionalised power. *Logic Journal of IGPL*, Oxford Univ Press, v. 4, n. 3, p. 427–443, 1996. ISSN 1367-0751.

OMICINI, A.; RICCI, A.; VIROLI, M. Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, v. 17, n. 3, p. 432–456, 2008.

PARR, T.; FISHER, K. Ll(*): the foundation of the antlr parser generator. In: *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*. New York, NY, USA: ACM, 2011. (PLDI '11), p. 425–436.

PIUNTI, M. *Designing and Programming Organizational Infrastructures for Agents situated in Artifact-based Environments*. Tese (Doutorado) — Universit‘ di Bologna, 2009.

PIUNTI, M. et al. Embodied organizations: a unifying perspective in programming agents, organizations and environments. In: FORNARA, N.; VOUROS, G. (Ed.). *Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN10@MALLOW)*. [S.l.]: CEUR, 2010. v. 627.

RICCI, A.; PIUNTI, M.; VIROLI, M. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, v. 23, n. 2, p. 158–192, 2011.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 2nd edition. ed. [S.l.]: Prentice-Hall, Englewood Cliffs, NJ, 2003.

SEARLE, J. *The Construction of Social Reality*. [S.l.]: Free Press, 1995.

SINGH, M. P. A social semantics for agent communications languages. In: *Proceedings of the Workshop on Agent Communication Languages, International Joint Conference on Artificial Intelligence (IJCAI-99*. [S.l.]: Springer, 2000.

STRATULAT, T.; FERBER, J.; TRANIER, J. Masq: towards an integral approach to interaction. In: *AAMAS (2)*. [S.l.: s.n.], 2009. p. 813–820.

WEYNS, D.; OMICINI, A.; ODELL, J. Environment, first-order abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, Citeseer, v. 14, n. 1, p. 5–30, 2007.

WOOLDRIDGE, M. *An introduction to multiagent systems*. [S.l.]: Oxford University Press New York, 1997.

APÊNDICE A – Considerações sobre a linguagem Jason

A sintaxe apresentada na seção 3.2.2 é inspirada em elementos de linguagens de programação baseadas em lógica, e, em especial, na linguagem de programação de agentes Jason (BORDINI; HÜBNER; WOOLDRIDGE, 2007). O interpretador da linguagem, inclusive, utiliza elementos desenvolvidos originalmente para o interpretador da linguagem Jason.

Neste apêndice, alguns aspectos da linguagem Jason são descritos. Esses aspectos podem ser necessários à compreensão da linguagem proposta na seção 3.2.2. A opção em tratar desses aspectos à parte da descrição da linguagem proposta justifica-se por tais aspectos não serem essenciais à compreensão da linguagem da mesma, apesar de auxiliarem à sua compreensão.

A.1 ELEMENTOS SINTÁTICOS ELEMENTARES

Para representar os elementos que compõem o mundo, à semelhança da linguagem Jason, são utilizados *termos*, que podem ser *constantes*, *variáveis* ou *estruturas* (conforme a Figura 32). Para representar um elemento específico do mundo, são utilizados *átomos* (que são um tipo específico de constante) ou *estruturas*. Um átomo é uma sequência de caracteres iniciada por letra minúscula. Por exemplo, `bob` é um átomo utilizado para representar uma pessoa chamada Bob. As estruturas, por sua vez, representam dados complexos, como, por exemplo, `aluno(``Tom'', 12435, idade(25), curso(inteligenciaArtificial))` pode representar informações sobre um aluno em uma instituição de ensino. Uma variável, representada por uma cadeia de caracteres iniciada por letra maiúscula, é um elemento que passa a se referir a um elemento específico no mundo no momento em que é unificada com uma *constante*, que pode ser um átomo, um número ou uma *string*.

Um outro elemento importante para a representação da informação em linguagens baseadas em lógica são os *predicados*, que indicam propriedades dos elementos do mundo (por exemplo, *alto(bob)*) ou relações entre elementos (por exemplo, *maior(bob,tom)*. É possível observar que a "forma" de um predicado é igual à de uma estrutura, sendo a diferença entre os dois puramente semântica: uma estrutura é utilizada para representar um elemento específico existente no mundo enquanto um predicado representa uma proposição lógica que pode ser usada como uma *fórmula* a respeito da qual é possível determinar um valor verdade(BORDINI; HÜBNER; WOOLDRIDGE, 2007). Dessa

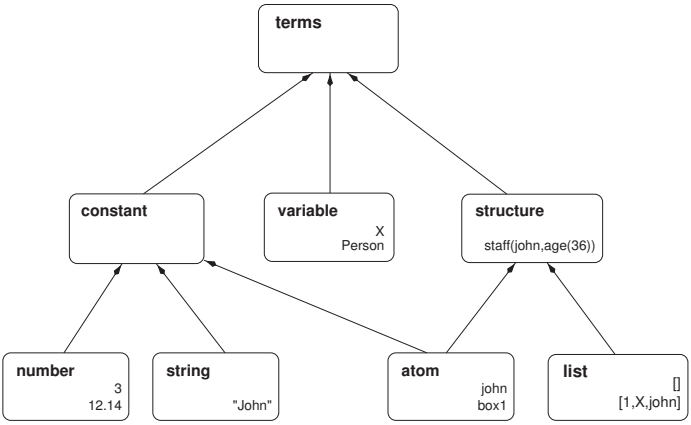


Figura 32 – Termos (BORDINI; HÜBNER; WOOLDRIDGE, 2007)

forma, enquanto a estrutura `aluno(tom, 112233, idade(25), curso(inteligenciaArtificial))` descreve um aluno chamado Tom, o predicado `inteligente(tom)` indica que Tom é inteligente e o predicado `amigo(tom,bob)` indica que Tom é amigo de Bob. Uma *fórmula* pode ser formada por um único predicado ou por um conjunto de predicados entre os quais existem relações representadas por operadores lógicos *e* (na linguagem representado pelo operador "&"), *ou* (na linguagem representado pelo operador "||") ou *não* (na linguagem representado pelo operador "`not`"). Considere-se, por exemplo, um mundo em que sejam verdadeiras as propriedades `alto(bob)` e `baixo(tom)`. Com relação a esse mundo, a Tabela 6 exemplifica algumas fórmulas e seus valores verdade.

As *variáveis* são elementos representados por uma cadeia de caracteres iniciada por letra maiúscula. Esses elementos podem ter valores associados a si através da operação de *substituição*, permitindo assim, que a linguagem execute uma operação de *unificação* quando necessário. A ideia por trás dos processos de substituição e unificação do Jason estão presentes em outras linguagens de programação baseadas em lógica (BITTENCOURT, 2001).

Um elemento incorporado à linguagem proposta neste trabalho, presente na linguagem Jason, mas não existente nas linguagens de programação baseadas em lógica tradicionais são as *anotações*. Esses ele-

Tabela 6 – Exemplos de fórmulas e valores verdade

Proposição	Valor Verdade
alto(bob)	verdadeiro
baixo(bob)	falso
notbaixo(bob)	verdadeiro
alto(bob)&alto(tom)	falso
alto(bob)‖alto(tom)	verdadeiro

mentos provém uma maneira de incorporar informações adicionais a elementos como predicados ou estruturas. As anotações são formadas por literais colocados entre colchetes. Um exemplo de predicado com anotação é :

`aula(inteligenciaArtificial, em_Andamento)[encerra("12pm")]`